NASA Technical Memorandum 102453

GRID2D/3D—A Computer Program for Generating Grid Systems in Complex-Shaped Two- and Three-Dimensional Spatial Domains

Part 1: Theory and Method

T.I-P. Shih
Carnegie-Mellon University
Pittsburgh, Pennsylvania

and

R.T. Bailey
University of Florida
Gainesville, Florida

and

H.L. Nguyen and R.J. Roelke
Lewis Research Center
Cleveland, Ohio

March 1990

NASA

CONTENTS

GRID2D/3D – A COMPUTER PROGRAM FOR GENERATING GRID SYSTEMS IN COMPLEX-SHAPED TWO- AND THREE-DIMENSIONAL SPATIAL DOMAINS PART 1: THEORY AND METHOD

T. I-P. Shih
Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

R. T. Bailey
Department of Mechanical Engineering
University of Florida
Gainesville, Florida 32611

H. L. Nguyen and R. J. Roelke
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

1.0 INTRODUCTION

Finite-difference (FD) and finite-volume (FV) methods are very powerful techniques for obtaining solutions to partial differential equations that govern fluid flow problems. However, in order to use these methods, it is necessary to replace the spatial domain of the problem being studied by a finite number of discrete points known as grid points. The process of replacing a spatial domain by a system of grid points is referred to as grid generation. Grid generation is a very important part of FD and FV methods because the system of grid points used strongly affects the accuracy, efficiency, and ease with which these methods generate solutions. In some instances, the ability or inability to generate an "acceptable" grid system determines whether FD or FV methods can or cannot be used.

Even though tremendous advances have been made in grid generation techniques during the past fifteen years (refs. 1 to 10), the generation of acceptable grid systems for geometrically complex three-dimensional spatial domains remains a difficult problem. Recently, a very efficient and versatile computer program, called GRID2D/3D, has been developed which can generate grid systems inside complex-shaped two- and three-dimensional (2- and 3-D) spatial domains. GRID2D/3D is so efficient that it is configured to run on PCs or PC compatible computers, though it can also be used on workstations and mainframes. The high efficiency of GRID2D/3D makes it especially useful for spatial domains that deform with time. This is because for such spatial domains, a different grid system must be generated at each time level, and the number of time levels can be thousands or more. This technical memorandum describes the theory and method behind GRID2D/3D and the types of grid systems that it can generate. Part 2 of this technical memorandum (to be published under a separate cover) will contain the program, GRID2D/3D, and a user's manual.

1.1 Types of Grid Systems That Can Be Generated by GRID2D/3D

Eiseman and Erlebacher (ref. 9) classified all possible grid systems that can be used by FD and FV methods as follows. At the broadest level, a grid system can be classified as structured, unstructured, or mixed depending upon how the grid points are connected to each other (fig. 1-1). A structured grid system, in turn, can be classified as a single grid or a composite grid. A single grid is one that is based on a single boundary-fitted coordinate system, whereas a composite grid is made up of two or more single grids patched together with each single grid having a different boundary-fitted coordinate system. Depending upon how the different single grids are patched together, a composite grid can further be classified as completely discontinuous, partially discontinuous, partially continuous, or completely continuous

(fig. 1-2). The continuity or discontinuity referred to here is concerned with that of the different boundary-fitted coordinate systems at locations where they are patched together in a composite grid.

Of the grid systems mentioned above, the unstructured grid system is the most versatile and the easiest to generate, especially for complicated-shaped spatial domains. But, the use of unstructured grid systems with FD and FV methods is still at a state of development (refs. 11 to 16). Presently, FD and FV methods almost exclusively use structured grid systems, and that is the type of grid system GRID2D/3D generates.

When a structured grid system is used with a FD or a FV method to obtain solutions to fluid flow problems, the structured grid system generated by GRID2D/3D or any other computer program should satisfy a number of conditions and they are

1. The total number of grid points in the grid system should be kept to the minimum needed for the FD or FV method to yield solutions of the desired accuracy. This condition is important for computational efficiency and can be achieved by clustering grid points in regions where they are needed (e.g., regions where gradients of the flow are large) and scattering them elsewhere.
2. One set of grid lines (coordinate lines of the boundary-fitted coordinate system) always should coincide with the boundary of the spatial domain regardless of the geometric complexity or motion of that boundary (i.e., the grid system should be boundary conforming). This condition is important because it enables FD and FV methods to implement boundary conditions easily and accurately for geometrically complex and/or deforming spatial domains.
3. Grid lines that intersect a boundary should intersect that boundary perpendicularly so that derivative boundary conditions can be implemented more easily and accurately. At the interior of the spatial domain, the angle

of intersection between grid lines only needs to be nearly orthogonal (i.e., between 45 and 135 degrees).

4. The spacings between grid points should change slowly from a region where grid points are concentrated to a region where grid points are sparsely distributed, especially in regions where gradients of the flow are large. This condition is important because Fourier components which make up the solution reflect and refract at interfaces where grid spacings change.
5. One set of grid lines should align with the flow direction. This condition is important for convection dominated flows when the aspect ratio of the control volume about each grid point is very high and/or when the thin layer Navier-Stokes equations are used to study such flows.

For complicated 2- and 3-D flows within geometrically complex spatial domains, the flow field varies considerably from one region to another. Thus, it is usually not possible to generate a single grid that would satisfy all of the above conditions at every part of the spatial domain. For such fluid flow problems, it is often necessary to generate a number of different single grids, each of which satisfies the above five conditions at a different part of the spatial domain. These single grids are then patched together to form a composite grid.

As noted earlier, depending upon how the different single grids are patched together, a composite grid can be completely discontinuous, partially discontinuous, partially continuous or completely continuous. In general, the more discontinuous a composite grid is, the easier it is to generate that grid and the harder it is to use that grid to obtain solutions.

Since the computer program, GRID2D/3D, is capable of generating single grids as well as the different types of composite grids, we briefly discuss below the advantages and disadvantages of the various types of composite grids in order to know when a specific type should be used.

1.1.1 Completely Discontinuous Composite Grids. – The major advantage of completely discontinuous composite grids, such as the chimera grid (refs. 17 to 21 and fig. 1-2(a)), is that they are the easiest to generate. To illustrate how chimera grids are generated, consider the spatial domain for the flow past an entire aircraft. To generate a chimera grid for such a spatial domain, all one has to do is generate a series of single grids, one about each component of the aircraft; for example one about the fuselage, another about the wing, still another about the nacelle, and so on. The patching process simply involves laying each single grid over the appropriate component of the aircraft, deciding the amount of overlap of different single grids, and ensuring that the entire spatial domain is filled with grid points. Since the geometry for each single grid can be made relatively simple and patching is trivial, the grid generation process is straightforward.

Another important advantage of this type of grid is that the structure of each single grid can be different from each other; for example, one single grid may have a C-C structure, while another may have an O-O or an O-H structure. Thus, it is possible to optimize each single grid for a different part of the spatial domain.

Still another important advantage of this type of grid is that it is the easiest to do local grid refinement. For a chimera grid, one can refine the grid at any location by simply generating a very fine single grid and then overlaying it wherever desired.

Also, this type of grid can easily be applied to problems in which one or more objects are moving relative to another object, such as the launching of missiles from an aircraft (ref. 20). For such problems, the completely discontinuous composite grid may be the best type of grid to use.

Finally, since composite grids are composed of a series of single grids, it is possible to do computations on one single grid at a time. This will reduce computer memory requirements considerably since only information on one single grid needs to

reside in the computer at any one time. This advantage is shared by all composite grids, continuous or discontinuous.

The major disadvantage of completely discontinuous composite grids is that it is more difficult to obtain solutions on such grids when using FD and FV methods than the other types of composite grids. This is because interpolation and averaging schemes are needed to transfer information from one single grid to another when and wherever two or more single grids overlap (refs. 18 to 21). Also, the schemes must be developed to ensure that boundary conditions are implemented correctly for the entire problem and that certain properties, such as the conservative and the transportive properties, are maintained in regions where two or more single grids overlap.

1.1.2 Partially Discontinuous Composite Grids. — Partially discontinuous composite grids (fig. 1-2(b)) are generated in the following manner. First, the spatial domain of the problem being studied is partitioned into a number of nonoverlapping, contiguous zones or blocks. Next, a single grid is generated within each zone. Finally, patching of the single grids simply involves putting each of the single grids into its respective zone.

Thus, partially discontinuous composite (PDC) grids differ from completely discontinuous composite (CDC) grids in that the single grids of PDC grids do not overlap each other. However, PDC and CDC grids have two important similarities. First, each single grid in both cases can have a structure that is different from each other. Second, the number of grid points in each single grid can be different from each other. Because of these two important similarities, the major advantages of PDC grids are very similar to those of the CDC grids. However, since single grids in a PDC grid do not overlap each other, PDC grids are somewhat more difficult to generate but are easier to use than CDC grids.

Examples of PDC grids include the "zonal" or "patched" grids (refs. 22 to 25). For such grids, the main difficulty is the implementation of boundary conditions at the interfaces where the different single grids of the PDC grid meet. References 22 to 25 describe a procedure for implementing such boundary conditions in a way that would ensure maintenance of the conservative property.

1.1.3 Partially and Completely Continuous Composite Grids. — Completely continuous composite (CCC) grids are grid systems in which all grid lines (i.e., coordinate lines of the boundary-fitted coordinate system) and all of their derivatives of every order are continuous at all interfaces where different single grids meet. In general, it is not necessary to construct CCC grids. Typically, FD and FV methods only require continuity of the grid lines and their first and, occasionally, second-order derivatives at the interfaces where different single grids meet. Composite grid systems with this limited degree of continuity are referred to as partially continuous composite (PCC) grids.

Figure 1-2(c) shows a PCC grid in which grid lines are all continuous, but first-order derivatives of the grid lines have discontinuities. It can readily be seen in that figure that the slope of the grid lines and the spacing between the grid lines change suddenly at the interface where the two single grids meet. Figure 1-2(d) shows a PCC grid in which the grid lines and their first-order derivatives are continuous everywhere including the interface where the two single grids meet. Such PCC grids have the same appearance as CCC grids.

The major advantage of PCC grids of the type shown in figure 1-2(d) is that this is the easiest grid system for FD and FV methods to use. This is because boundary conditions can be implemented easily at interfaces where different single grids meet. In fact, it is not even necessary to treat the interfaces where different single grids meet as boundaries since computations can be carried across them. For PCC

grids, the complete spatial domain of the problem can be mapped onto a single transformed domain, even though different boundary-fitted coordinate systems have been used in different parts of the spatial domain (fig. 1-3).

Here, it is important to note that not all grid systems which appear to be continuous are continuous. Figure 1-4 shows a composite grid that appears to be continuous but belongs to the PDC grids because it is impossible to map the entire spatial domain onto one transformed domain.

The major disadvantage of PCC grids is that they are the most difficult to generate when compared to CDC and PDC grids. Another disadvantage of PCC grids is that the structure and number of grid points in each single grid must satisfy certain compatibility conditions to ensure continuity. These compatibility conditions make it more difficult to optimize each single grid for a specific area. It also makes it more difficult to do local grid refinement.

Thus, there are many spatial domains for which it is extremely difficult, if not impossible, to generate a PCC grid that is acceptable. However, when it is possible, then the generation of such grids is worthwhile because of the ease with which they can be used. References 26-31 show a number of examples of how to construct PCC grids for complex-shaped 2- and 3-D spatial domains.

1.2 Grid Generation Methods Used in GRID2D/3D and Why

In the previous section, we discussed the various types of grid systems that can be generated by GRID2D/3D and when a specific type should be used. In this section, we briefly outline the different types of advanced grid generation techniques and give reasons why GRID2D/3D is based on one class of methods.

All grid generation techniques can be divided into two major classes – differential equation methods and algebraic methods. Differential equation methods

generate grid systems by solving a system of partial differential equations (PDEs) which describes how grid points are to be distributed within the spatial domain. Examples of differential equation methods include methods based on elliptic PDEs (e.g., harmonic mapping, 2- and 3-D quasilinear systems), methods based on hyperbolic PDEs (e.g., orthogonal trajectories and field methods), and methods based on parabolic PDEs (refs. 2, 3, and 8). Most of these methods require a significant amount of computational effort since the systems of PDEs that must be solved are quasilinear and often as complicated as the PDEs that govern the fluid flow problem. This is especially true when using these methods to generate grid systems in 3-D spatial domains and in spatial domains that deform with time.

Algebraic methods generate grid systems by interpolating between boundaries of the spatial domain. Since no PDE needs to be solved in the grid generation process, algebraic grid generation methods are computationally much more efficient than differential equation methods. Examples of algebraic grid generation methods include shearing transformations (ref. 32) and transfinite interpolation methods (refs. 33 to 35). Transfinite interpolation methods include the Two-Boundary Method (refs. 36 and 37), Four-Boundary Method (refs. 38 and 39), Six-Boundary Method (refs. 39 and 40), and multisurface methods (refs. 41 to 43).

Whether one uses a differential equation method or an algebraic method, the grid generation process is always iterative. This is because the "acceptable" grid system is arrived at via trial and error after generating a series of unsatisfactory grids. The effort of the iterative process is, of course, compounded many times for spatial domains which can deform with time since, for such domains, a different grid system is needed for each time level and the number of time levels can be thousands or more. Hence, the efficiency of the grid generation process is extremely important for problems with 3-D spatial domains and for problems in which the spatial domain can deform.

Since GRID2D/3D is intended for complex-shaped 2- and 3-D spatial domains and for spatial domains that can deform with time, GRID2D/3D generates grid systems by using algebraic grid generation methods. Depending upon the complexity and dimensionality of the spatial domain, GRID2D/3D uses one of the following three methods, all of which are very similar and are based on transfinite interpolation: the Two-Boundary Method, the Four-Boundary Method, and the Six-Boundary Method. These methods were chosen because of their high efficiency and their ability to provide very precise controls over the distribution of grid points in the spatial domain when used in conjunction with stretching functions (refs. 44 and 45). Also, these methods can generate grid lines that intersect boundaries orthogonally.

By using these algebraic grid generation methods, GRID2D/3D generates single grids with grid lines that are continuous and differentiable everywhere up to the second-order. GRID2D/3D generates composite grids by patching together two or more single grids. The patching can be discontinuous or continuous. For continuous composite grids, the grid lines are continuous and differentiable everywhere up to the second-order except at interfaces where different single grids meet. At interfaces where different single grids meet, the grid lines are only differentiable up to the first-order.

In order to use the Two-, Four-, and Six-Boundary Methods to generate grid systems, the boundaries of the spatial domains must be represented mathematically in parametric form. This is a difficult problem for complicated-shaped spatial domains because the boundaries of such domains are complicated as well. In GRID2D/3D, parametric equations for boundary curves of 2-D spatial domains are generated by either spline interpolation (ref. 46) or tension-spline interpolation (ref. 47). Parametric equations for boundary surfaces of 3-D spatial domains can be generated by a number of techniques including linear Coons' interpolation (ref. 33), bidirectional spline interpolation (refs. 48 and 49), and bihyperbolic spline interpolation (ref. 50). In GRID2D/3D, parametric equations for these 3-D surfaces are generated by either

linear Coon's interpolation, bihyperbolic spline interpolation, or a new technique referred to as 3-D bidirectional Hermite interpolation.

The details of the algebraic grid generation methods used in GRID2D/3D are given in Sections 2.0 and 3.0. Examples of single and composite grids generated by GRID2D/3D are given in Section 4.0. A listing of the computer program, GRID2D/3D, and a user's manual will be found in Part 2 of this technical memorandum which will be published under a separate cover.

2.0 THE TWO-, FOUR-, AND SIX-BOUNDARY METHODS OF ALGEBRAIC GRID GENERATION

As noted in Section 1.0, GRID2D/3D generates grid systems by using one of the following three algebraic grid generation methods: the Two-Boundary Method, the Four-Boundary Method, and the Six-Boundary Method. All three of these methods are based on a general technique known as transfinite interpolation (refs. 33 and 35). The Two-Boundary Method, described by Smith (ref. 36) and Yang and Shih (ref. 37), is intended for problems in which it is only necessary to map correctly two arbitrary-shaped boundaries of the spatial domain. For problems in which it is necessary to map correctly all of the boundaries of the spatial domain or at least more than two of them, then the Four-Boundary Method or the Six-Boundary Method, described by Vinokur and Lombard (ref. 38), Rizzi and Eriksson (ref. 39), and Eriksson (ref. 40), can be used. All three of these methods can generate grid systems in 3-D spatial domains; the Two- and Four-Boundary Methods can also generate grid systems in 2-D spatial domains.

In this section, the details of these methods are described by using each of them to generate either a 2-D or a 3-D grid system. Our step-by-step descriptions of the methods follow closely those of Yang and Shih (ref. 37) and Shih (ref. 51). It is our intention to present the methods in a clear manner so that the reader might easily implement any one of the three methods.

2.1 The Two-Boundary Method

As mentioned previously, the Two-Boundary Method is intended for problems in which it is only necessary to map correctly two arbitrary-shaped boundaries of the spatial domain.

There are a number of problems for which this holds (Section 6-4-3 of ref. 51); a few example problems with such spatial domains are shown in figure 2-1.

Here, we note that the Two-Boundary Method can correctly map all of the boundaries of a spatial domain if the remaining boundaries are straight lines in the 2-D case or flat surfaces in the 3-D case. Figures 2-1(a) and 2-1(d) illustrate these cases.

The Two-Boundary Method involves the following eight major steps (refs. 37 and 51):

1. Define the nature of the coordinate transformation.
2. Select a time-stretching function.
3. Select the two boundaries of the spatial domain that must be mapped correctly. These two boundaries cannot touch each other at any point.
4. Describe the two boundaries selected in Step 3 in parametric form.
5. Define curves that connect the two boundaries using transfinite interpolation.
6. Discretize the domain (i.e., replace the continuous domain of the problem by time levels and grid points).
7. Control the distribution of the grid points with stretching functions.
8. Calculate the metric coefficients needed by the FD or the FV method to obtain solutions.

The details of these eight steps of the Two-Boundary Method are described below by generating a grid system in a 3-D, deforming spatial domain shown in figure 2-2(a) for the problem of compressible flow through a converging-diverging channel. The spatial domain of interest is the region bounded by surfaces 1 through 6 in figure 2-2(a). The spatial domain deforms because surfaces 1 and 2 deform with time.

Step 1 - Define the Coordinate Transformation. The first step of the Two-Boundary Method is to define the coordinate transformation between the coordinate system of the spatial domain and the boundary-fitted coordinate system of the transformed domain. For

3-D spatial domains in which grid points are allowed to move, grid generation involves the determination of the following coordinate transformation:

$$(x,y,z,t) \leftrightarrow (\xi,\eta,\zeta,\tau) \tag{2.1a}$$

or, more specifically,

$$t = t(\tau) \tag{2.1b}$$

$$x = x(\xi,\eta,\zeta,\tau) \tag{2.1c}$$

$$y = y(\xi,\eta,\zeta,\tau) \tag{2.1d}$$

$$z = z(\xi,\eta,\zeta,\tau) \tag{2.1e}$$

where x, y, z, and t represent the coordinate system of the spatial domain and ξ, η, ζ, and τ represent the boundary-fitted coordinate system of some transformed domain (fig. 2-2).

Step 2 - Select a Time-Stretching Function. The next step is to define a relationship between t and τ. For our example, we set t equal to τ; that is,

$$t = \tau \tag{2.2}$$

Thus, no time-stretching function is used. Time stretching may be useful when variable time-step sizes are used with FD or FV schemes that involve information at more than two time levels.

Step 3 - Select Two Boundaries of the Spatial Domain. The third step is to select the two boundaries of the spatial domain that are to be mapped correctly. These two boundaries must not intersect each other at any point. For the spatial domain of figure 2-2(a), we select boundary surfaces 1 and 2.

Since ξ, η, ζ and τ represent a boundary-fitted coordinate system, boundary surfaces of the spatial domain in the x-y-z-t coordinate system must correspond to coordinate planes in the ξ-η-ζ-τ coordinate system. We choose surfaces 1 and 2 to correspond to coordinate planes $\eta = 0$ and $\eta = 1$, respectively (fig. 2-2); that is,

$$X_1 = x(\xi, \eta=0, \zeta, \tau) = X_1(\xi,\zeta,\tau) \tag{2.3a}$$

$$Y_1 = y(\xi, \eta=0, \zeta, \tau) = Y_1(\xi,\zeta,\tau) \tag{2.3b}$$

$$Z_1 = z(\xi, \eta=0, \zeta, \tau) = Z_1(\xi,\zeta,\tau) \tag{2.3c}$$

and

$$X_2 = x(\xi, \eta=1, \zeta, \tau) = X_2(\xi,\zeta,\tau) \tag{2.4a}$$

$$Y_2 = y(\xi, \eta=1, \zeta, \tau) = Y_2(\xi,\zeta,\tau) \tag{2.4b}$$

$$Z_2 = z(\xi, \eta=1, \zeta, \tau) = Z_2(\xi,\zeta,\tau) \tag{2.4c}$$

Here, X_1, Y_1, and Z_1 are the x-, y-, and z-coordinates of surface 1, and X_2, Y_2, and Z_2 are the x-, y-, and z-coordinates of surface 2. The remaining four boundaries -- surfaces 3, 4, 5, and 6 -- are mapped to coordinate planes $\xi = 0$, $\xi = 1$, $\zeta = 0$, and $\zeta = 1$, respectively (fig. 2-2).

Step 4 - Describe the Two Boundaries Selected in Parametric Form. Once the two boundaries have been selected, the next step is to represent these two boundaries in parametric form as suggested by the form of equations (2.3) and (2.4). Equations (2.3) and (2.4) also tell us that the three parameters which must be used to describe surfaces 1 and 2 are ξ, ζ, and τ. Keep in mind that had we chosen different coordinate planes in the transformed domain for our boundaries, these parameters could have been different.

In this example, we assume that the parametric equations describing surfaces 1 and 2 are given and they are

$$X_1 = \xi \; / \; L_x \tag{2.5a}$$

$$Y_1 = A \sin(w\tau) \, [1 - \cos(2 \pi \xi)] \, [1 - \cos(2 \pi \zeta)] \tag{2.5b}$$

$$Z_1 = \zeta \; / \; L_y \tag{2.5c}$$

and

$$X_2 = \xi \; / \; L_X \tag{2.6a}$$

$$Y_2 = L_y - A \sin(w\tau) \; [1 - \cos(2 \; \pi \; \xi)] \; [1 - \cos(2 \; \pi \; \zeta)] \tag{2.6b}$$

$$Z_2 = \zeta \; / \; L_y \tag{2.6c}$$

where A, w, L_X, L_y, and L_Z are given constants.

For most problems, parametric equations describing the boundaries of spatial domains are not given. What is usually given are sets of coordinates which describe the positions of a finite number of discrete points located on the boundary, and numerical methods must be used to generate the required parametric equations. The numerical methods used in GRID2D/3D for this purpose are described in Section 3.0.

Step 5 - Define Curves Connecting Boundaries Using Transfinite Interpolation. A number of different transfinite interpolation techniques can be used to derive curves which connect the two boundaries chosen in Step 3. Here, we consider two such methods: transfinite interpolation based on Lagrange interpolation and transfinite interpolation based on Hermite interpolation.

Transfinite interpolation based on Lagrange interpolation is also known as linearly blended transfinite interpolation. When this technique is used to generate connecting curves between surfaces 1 and 2, the resulting curves have the following functional form:

$$x(\xi,\eta,\zeta,\tau) = X_1(\xi,\zeta,\tau) \;\; l_1(\eta) + X_2(\xi,\zeta,\tau) \;\; l_2(\eta) \tag{2.7a}$$

$$y(\xi,\eta,\zeta,\tau) = Y_1(\xi,\zeta,\tau) \;\; l_1(\eta) + Y_2(\xi,\zeta,\tau) \;\; l_2(\eta) \tag{2.7b}$$

$$z(\xi,\eta,\zeta,\tau) = Z_1(\xi,\zeta,\tau) \;\; l_1(\eta) + Z_2(\xi,\zeta,\tau) \;\; l_2(\eta) \tag{2.7c}$$

where X_1, Y_1, and Z_1 given by equation (2.5) describe surface 1, and X_2, Y_2, and Z_2 given by equation (2.6) describe surface 2. The functions l_1 and l_2 are blending functions, and, for the linear case, they have the functional form of first degree Lagrange interpolating polynomials.

As such, they connect two points – one on each surface having the same ξ, ζ, and τ values – and are constrained by the following expressions:

$$l_1(\eta = 0) = 1 \qquad l_1(\eta = 1) = 0$$
$$l_2(\eta = 0) = 0 \qquad l_2(\eta = 1) = 1$$

With these constraints, l_1 and l_2 become

$$l_1(\eta) = 1 - \eta \tag{2.8a}$$
$$l_2(\eta) = \eta \tag{2.8b}$$

Substitution of equation (2.8) into equation (2.7) yields the desired linear connecting curves

$$x(\xi,\eta,\zeta,\tau) = \mathrm{X}_1(\xi,\zeta,\tau)\,(1-\eta) + \mathrm{X}_2(\xi,\zeta,\tau)\,\eta \tag{2.9a}$$
$$y(\xi,\eta,\zeta,\tau) = \mathrm{Y}_1(\xi,\zeta,\tau)\,(1-\eta) + \mathrm{Y}_2(\xi,\zeta,\tau)\,\eta \tag{2.9b}$$
$$z(\xi,\eta,\zeta,\tau) = \mathrm{Z}_1(\xi,\zeta,\tau)\,(1-\eta) + \mathrm{Z}_2(\xi,\zeta,\tau)\,\eta \tag{2.9c}$$

It is often desirable for connecting curves to intersect boundaries orthogonally so that derivative boundary conditions can be implemented accurately. Since the curves described by equation (2.9) are straight lines, they will not, in general, intersect boundaries orthogonally. One way to remedy this is to use transfinite interpolation based on Hermite interpolation to form the connecting curves. Transfinite interpolation based on Hermite interpolation allows specification of the derivatives at the end points of the curves, enabling one to force orthogonality at the boundary. When this method is used to generate connecting curves between surfaces 1 and 2, the resulting cubic curves have the following functional form:

$$x(\xi,\eta,\zeta,\tau) = X_1(\xi,\zeta,\tau)\, h_1(\eta) + X_2(\xi,\zeta,\tau)\, h_2(\eta)$$

$$+ \frac{\partial x(\xi,\eta=0,\zeta,\tau)}{\partial \eta}\, h_3(\eta) + \frac{\partial x(\xi,\eta=1,\zeta,\tau)}{\partial \eta}\, h_4(\eta) \tag{2.10a}$$

$$y(\xi,\eta,\zeta,\tau) = Y_1(\xi,\zeta,\tau)\, h_1(\eta) + Y_2(\xi,\zeta,\tau)\, h_2(\eta)$$

$$+ \frac{\partial y(\xi,\eta=0,\zeta,\tau)}{\partial \eta}\, h_3(\eta) + \frac{\partial y(\xi,\eta=1,\zeta,\tau)}{\partial \eta}\, h_4(\eta) \tag{2.10b}$$

$$z(\xi,\eta,\zeta,\tau) = Z_1(\xi,\zeta,\tau)\, h_1(\eta) + Z_2(\xi,\zeta,\tau)\, h_2(\eta)$$

$$+ \frac{\partial z(\xi,\eta=0,\zeta,\tau)}{\partial \eta}\, h_3(\eta) + \frac{\partial z(\xi,\eta=1,\zeta,\tau)}{\partial \eta}\, h_4(\eta) \tag{2.10c}$$

where X_1, Y_1, and Z_1 given by equation (2.5) describe surface 1, and X_2, Y_2, and Z_2 given by equation (2.6) describe surface 2. The functions h_1, h_2, h_3, and h_4 are blending functions which connect two points – one on each surface having the same ξ, ζ, and τ values. They are constrained by the following expressions:

$$h_1(\eta = 0) = 1 \qquad h_1(\eta = 1) = 0$$

$$\frac{\partial h_1(\eta = 0)}{\partial \eta} = 0 \qquad \frac{\partial h_1(\eta = 1)}{\partial \eta} = 0$$

$$h_2(\eta = 0) = 0 \qquad h_2(\eta = 1) = 1$$

$$\frac{\partial h_2(\eta = 0)}{\partial \eta} = 0 \qquad \frac{\partial h_2(\eta = 1)}{\partial \eta} = 0$$

$$h_3(\eta = 0) = 0 \qquad h_3(\eta = 1) = 0$$

$$\frac{\partial h_3(\eta = 0)}{\partial \eta} = 1 \qquad \frac{\partial h_3(\eta = 1)}{\partial \eta} = 0$$

$$h_4(\eta = 0) = 0 \qquad h_4(\eta = 1) = 0$$

$$\frac{\partial h_4(\eta = 0)}{\partial \eta} = 0 \qquad \frac{\partial h_4(\eta = 1)}{\partial \eta} = 1$$

With these constraints, h_1, h_2, h_3, and h_4 become (ref. 36)

$$h_1(\eta) = 2\eta^3 - 3\eta^2 + 1 \tag{2.11a}$$

$$h_2(\eta) = -2\eta^3 + 3\eta^2 \tag{2.11b}$$

$$h_3(\eta) = \eta^3 - 2\eta^2 + \eta \tag{2.11c}$$

$$h_4(\eta) = \eta^3 - \eta^2 \tag{2.11d}$$

We choose the values for $\frac{\partial x(\xi,\eta=0,\zeta,\tau)}{\partial \eta}$, $\frac{\partial x(\xi,\eta=1,\zeta,\tau)}{\partial \eta}$, $\frac{\partial y(\xi,\eta=0,\zeta,\tau)}{\partial \eta}$, and $\frac{\partial y(\xi,\eta=1,\zeta,\tau)}{\partial \eta}$ so that the connecting curves given by equation (2.10) will intersect surfaces 1 and 2 orthogonally. For surface 1, this will occur when the cross product of $\tilde{n}$ (a vector normal to surface 1) and $\tilde{e}_\eta$ (the vector tangent to the connecting curve) is zero. This will be the case when

$$\frac{\partial x(\xi,\eta=0,\zeta,\tau)}{\partial \eta} = K_1(\xi,\zeta,\tau)\left(\frac{\partial Y_1}{\partial \zeta}\frac{\partial Z_1}{\partial \xi} - \frac{\partial Z_1}{\partial \zeta}\frac{\partial Y_1}{\partial \xi}\right) \tag{2.12a}$$

$$\frac{\partial y(\xi,\eta=0,\zeta,\tau)}{\partial \eta} = -K_1(\xi,\zeta,\tau)\left(\frac{\partial X_1}{\partial \zeta}\frac{\partial Z_1}{\partial \xi} - \frac{\partial Z_1}{\partial \zeta}\frac{\partial X_1}{\partial \xi}\right) \tag{2.12b}$$

$$\frac{\partial z(\xi,\eta=0,\zeta,\tau)}{\partial \eta} = K_1(\xi,\zeta,\tau)\left(\frac{\partial X_1}{\partial \zeta}\frac{\partial Y_1}{\partial \xi} - \frac{\partial Y_1}{\partial \zeta}\frac{\partial X_1}{\partial \xi}\right) \tag{2.12c}$$

Similarly, the connecting curves will intersect surface 2 orthogonally when

$$\frac{\partial x(\xi,\eta=1,\zeta,\tau)}{\partial \eta} = K_2(\xi,\zeta,\tau)\left(\frac{\partial Y_2}{\partial \xi}\frac{\partial Z_2}{\partial \zeta} - \frac{\partial Z_2}{\partial \xi}\frac{\partial Y_2}{\partial \zeta}\right) \tag{2.13a}$$

$$\frac{\partial y(\xi,\eta=1,\zeta,\tau)}{\partial \eta} = -K_2(\xi,\zeta,\tau)\left(\frac{\partial X_2}{\partial \xi}\frac{\partial Z_2}{\partial \zeta} - \frac{\partial Z_2}{\partial \xi}\frac{\partial X_2}{\partial \zeta}\right) \tag{2.13b}$$

$$\frac{\partial z(\xi,\eta=1,\zeta,\tau)}{\partial \eta} = K_2(\xi,\zeta,\tau)\left(\frac{\partial X_2}{\partial \xi}\frac{\partial Y_2}{\partial \zeta} - \frac{\partial Y_2}{\partial \xi}\frac{\partial X_2}{\partial \zeta}\right) \tag{2.13c}$$

$K_1(\xi,\zeta,\tau)$ and $K_2(\xi,\zeta,\tau)$ in equations (2.12) and (2.13) are known as the "K factors" and are chosen by trial and error so that no overlapping of the connecting curves takes place in the interior of the spatial domain. For our problem, the "K factors" are constants given by

$$K_1(\xi,\zeta,\tau) = K_2(\xi,\zeta,\tau) = 0.2$$

The values of $\frac{\partial X_1(\xi,\zeta,\tau)}{\partial \xi}$, $\frac{\partial Y_1(\xi,\zeta,\tau)}{\partial \xi}$, $\frac{\partial X_2(\xi,\zeta,\tau)}{\partial \xi}$, and $\frac{\partial Y_2(\xi,\zeta,\tau)}{\partial \xi}$ in equations (2.12) and (2.13) can easily be found by analytically differentiating equations (2.5) and (2.6) or by using finite-difference formulas. Thus, substitution of equations (2.11), (2.12), and (2.13) into equation (2.10) yields the desired cubic connecting curves based on Hermite interpolation.

Step 6 - Discretize the Domain. Steps 1 through 5 above describe how we map the x-y-z-t coordinate system onto the ξ-η-ζ-τ coordinate system. Having done this, we now need to discretize the domain in the ξ-η-ζ-τ coordinate system; that is, we need to replace the continuous domain by time levels and grid points.

For our problem, we replace the time domain by equally incremented time levels; that is,

$$\tau^n = n\,\Delta\tau\ ,\quad n = 0, 1, 2, \ldots \tag{2.14}$$

where τ^n denotes the time at time level n, and $\Delta\tau$ denotes the constant time-step size.

We also replace the spatial domain in the ξ-η-ζ-τ coordinate system by $IL \times JL \times KL$ equally spaced grid points (fig. 2-3(b)). The locations of these grid points are given by the ordered triples (ξ_i,η_j,ζ_k) where

$$\xi_i = (i-1)\,\Delta\xi\ ,\quad i = 1, 2, \ldots, IL \tag{2.15a}$$
$$\eta_j = (j-1)\,\Delta\eta\ ,\quad j = 1, 2, \ldots, JL \tag{2.15b}$$
$$\zeta_k = (k-1)\,\Delta\zeta\ ,\quad k = 1, 2, \ldots, KL \tag{2.15c}$$
$$\Delta\xi = \frac{1}{IL-1} \qquad \Delta\eta = \frac{1}{JL-1} \qquad \Delta\zeta = \frac{1}{KL-1} \tag{2.15d}$$

If we substitute equation (2.15) into equation (2.10), we can obtain the locations of the grid points in the x-y-z-t coordinate system. Figure 2-3(a) shows the system of grid points for

our problem in the x-y-z-t coordinate system obtained by using equations (2.10) and (2.15) at one value of τ.

Step 7 - Control the Distribution of Grid Points. At this point, we need to examine the grid system shown in figure 2-3(a) and ask, "Is the distribution of grid points satisfactory?" In order to answer this question, we need to consider the physics of the problem for which the grid is generated. For accurate solutions, grid points should be clustered in regions of the spatial domain where sharp gradients in the dependent variables exist. Such clustering can be achieved by the use of stretching functions (refs. 44 and 45).

For compressible flow within the 3-D spatial domain shown in figure 2-2(a), we expect steep gradients near the walls (surfaces 1, 2, 3, and 4). Grid points can be clustered near surfaces 1 and 2 by replacing η in equation (2.10) by the following stretching function expression:

$$\frac{(\beta_\eta+1)\Big((\beta_\eta+1)/(\beta_\eta-1)\Big)^{(2\eta\text{-}1)}-\beta_\eta+1}{2\Big\{1+(\beta_\eta+1)/(\beta_\eta-1)\Big)^{(2\eta\text{-}1)}\Big\}} \tag{2.16}$$

where β_η is a constant greater than unity. More clustering takes place in the η direction near $\eta=0$ and $\eta=1$ as β_η approaches unity.

In a similar manner, grid points can be clustered near surfaces 3 and 4 by replacing ξ in equation (2.10) by

$$\frac{(\beta_\xi+1)\Big((\beta_\xi+1)/(\beta_\xi-1)\Big)^{(2\xi\text{-}1)}-\beta_\xi+1}{2\Big\{1+(\beta_\xi+1)/(\beta_\xi-1)\Big)^{(2\xi\text{-}1)}\Big\}} \tag{2.17}$$

where β_ξ is a constant greater than unity that acts in the ξ direction as β_η does in the η direction.

The new distribution of grid points in the x-y-z-t coordinate system after stretching is shown in figure 2-4.

Step 8 - Calculate Metric Coefficients. Once we obtain a satisfactory distribution of grid points in the x-y-z-t coordinate system, we are ready to calculate the metric coefficients which are needed to obtain finite-difference solutions to the partial differential equations governing the problem. The metric coefficients appear in the governing equations when they are transformed from the coordinate system of the spatial domain (x-y-z-t in our example) to the boundary-fitted coordinate system of the transformed domain (ξ-η-ζ-τ). With the coordinate transformation described by equation (2.1), the metric coefficients which appear are τ_t, ξ_t, η_t, ζ_t, ξ_x, η_x, ζ_x, ξ_y, η_y, ζ_y, ξ_z, η_z, and ζ_z. These metric coefficients can be calculated using the following expressions (refs. 2, 3, and 8):

$$\tau_t = 1 \tag{2.18a}$$

$$\xi_x = (y_\eta z_\zeta - y_\zeta z_\eta)/J \tag{2.18b}$$

$$\xi_y = -(x_\eta z_\zeta - x_\zeta z_\eta)/J \tag{2.18c}$$

$$\xi_z = (x_\eta y_\zeta - x_\zeta y_\eta)/J \tag{2.18d}$$

$$\xi_t = -[\, x_\tau(y_\eta z_\zeta - y_\zeta z_\eta) - y_\tau(x_\eta z_\zeta - x_\zeta z_\eta) + z_\tau(x_\eta y_\zeta - x_\zeta y_\eta)\,]/J \tag{2.18e}$$

$$\eta_x = -(y_\xi z_\zeta - y_\zeta z_\xi)/J \tag{2.18f}$$

$$\eta_y = (x_\xi z_\zeta - x_\zeta z_\xi)/J \tag{2.18g}$$

$$\eta_z = -(x_\xi y_\zeta - x_\zeta y_\xi)/J \tag{2.18h}$$

$$\eta_t = [\, x_\tau(y_\xi z_\zeta - y_\zeta z_\xi) - y_\tau(x_\xi z_\zeta - x_\zeta z_\xi) + z_\tau(x_\xi y_\zeta - x_\zeta y_\xi)\,]/J \tag{2.18i}$$

$$\zeta_x = (y_\xi z_\eta - y_\eta z_\xi)/J \tag{2.18j}$$

$$\zeta_y = -(x_\xi z_\eta - x_\eta z_\xi)/J \tag{2.18k}$$

$$\zeta_z = (x_\xi y_\eta - x_\eta y_\xi)/J \tag{2.18l}$$

$$\zeta_t = -[\, x_\tau(y_\xi z_\eta - y_\eta z_\xi) - y_\tau(x_\xi z_\eta - x_\eta z_\xi) + z_\tau(x_\xi y_\eta - x_\eta y_\xi)\,]/J \tag{2.18m}$$

where J is the Jacobian given by

$$J = x_\xi(y_\eta z_\zeta - y_\zeta z_\eta) - x_\eta(y_\xi z_\zeta - y_\zeta z_\xi) + x_\zeta(y_\xi z_\eta - y_\eta z_\xi) \tag{2.18n}$$

The derivative terms x_ξ, x_η, x_ζ, y_ξ, y_η, y_ζ, z_ξ, z_η, and z_ζ in equation (2.18) can be evaluated either analytically by differentiating equation (2.10) or numerically by using finite-difference formulas. The correct way to evaluate the metric coefficients depends on how the governing equations written in the boundary-fitted coordinate system are cast. If the governing equations are cast in strong conservation-law form, then the metric coefficient must be evaluated numerically and the finite-difference formulas used to evaluate them must be the same as those used in the finite-difference method of solution. If the governing equations are cast in weak conservation-law form, then no conditions are imposed on the method for evaluating the metric coefficients themselves but there is a condition on how the derivatives of metric coefficients can be evaluated. Finally, if the chain-rule conservation-law form is used, then no conditions are imposed on how metric coefficients and their derivatives are evaluated. This important topic is addressed in references 51 to 54.

2.2 The Four-Boundary Method

The Four-Boundary Method for generating grid points is intended for situations where four boundaries of a spatial domain need to be mapped correctly from the spatial domain to the transformed domain. The Four-Boundary Method (refs. 38, 39, 40, and 51) is an extension of the Two-Boundary Method described in the previous section, and, as such, consists of the same eight major steps with minor variations, namely:

1. Define the nature of the coordinate transformation.

2. Select a time-stretching function.

3. Select the four boundaries of the spatial domain that must be mapped correctly.

4. Describe the four boundaries selected in Step 3 in parametric form.

5. Map the spatial domain to a transformed domain.

6. Discretize the domain (i.e., replace the continuous domain of the problem with time levels and grid points).

7. Control the distribution of the grid points with stretching functions.

8. Calculate the metric coefficients needed by the FD or the FV method to obtain solutions.

We will describe each of the eight steps of the Four-Boundary Method by generating a grid system inside the spatial domain shown in figure 2-5(a). This spatial domain is the cross-section of a steel bar whose temperature distribution we wish to determine. The bar has a uniform cross-section along its length, we shall consider only the 2-D region bounded by curves 1 through 4 (fig. 2-5(a)).

Step 1 - Define the Coordinate Transformation. For the 2-D, non-deforming spatial domain shown in figure 2-5(a), we seek a coordinate transformation of the form

$$(x,y,t) \leftrightarrow (\xi,\eta,\tau) \tag{2.19a}$$

or, more specifically,

$$t = t(\tau) \tag{2.19b}$$
$$x = x(\xi,\eta) \tag{2.19c}$$
$$y = y(\xi,\eta) \tag{2.19d}$$

Here, x, y, and t comprise the coordinate system of the spatial domain, and ξ, η, and τ comprise the boundary-fitted coordinate system of the transformed domain (figs. 2-5(a) and 2-5(b)).

Step 2 - Select a Time-Stretching Function. A time-stretching function is not used and t is set equal to τ as shown by equation (2.2).

Step 3 - Select Four Boundaries of the Spatial Domain. Since the spatial domain of figure 2-5(a) has only four boundaries, all four boundaries are selected. We choose curves 1, 2, 3, and 4 to correspond to coordinate lines $\eta=0$, $\eta=1$, $\xi=0$, and $\xi=1$, respectively (fig. 2-5); that is,

$$X_1 = x(\xi, \eta=0) = X_1(\xi) \tag{2.20a}$$
$$Y_1 = y(\xi, \eta=0) = Y_1(\xi) \tag{2.20b}$$
$$X_2 = x(\xi, \eta=1) = X_2(\xi) \tag{2.20c}$$
$$Y_2 = y(\xi, \eta=1) = Y_2(\xi) \tag{2.20d}$$
$$X_3 = x(\xi=0, \eta) = X_3(\eta) \tag{2.20e}$$
$$Y_3 = y(\xi=0, \eta) = Y_3(\eta) \tag{2.20f}$$
$$X_4 = x(\xi=1, \eta) = X_4(\eta) \tag{2.20g}$$
$$Y_4 = y(\xi=1, \eta) = Y_4(\eta) \tag{2.20h}$$

Here, X_1 and Y_1 are the x- and y-coordinates of curve 1; X_2 and Y_2 are the x- and y-coordinates of curve 2; X_3 and Y_3 are the x- and y-coordinates of curve 3; and X_4 and Y_4 are the x- and y-coordinates of curve 4.

Step 4 - Describe the Four Boundaries Selected in Parametric Form. Having selected four boundaries, we now need to represent these boundaries in parametric form as suggested by equation (2.20). For this problem, information about the four boundary curves is given in the form of a set of discrete points which lie along the curves. Thus, interpolation techniques must be used to generate the parametric equations of the boundary curves. In GRID2D/3D, either spline or tension spline interpolation can be used. Here, tension spline described in Section 3.0 is used to obtain parametric equations for the four curves in terms of the parameter ξ for curves 1 and 2 and in terms of η for curves 3 and 4. Figure 2-6 shows the curve approximations thus generated.

Step 5 - Map the Spatial Domain. The Four-Boundary Method maps the spatial domain to the transformed domain in two steps. The first step is essentially the same as Step 5 of the Two-Boundary Method in which we select two of the four boundaries which do not touch each other and which have been described parametrically using the same parameter (either ξ or η). As in Step 5 of the Two-Boundary Method, curves that connect these two boundaries are specified by using Hermite transfinite interpolation. When this step is completed, the two boundaries that were selected will be mapped correctly, but the other two boundaries will in general be mapped incorrectly. To remedy this, a second step is performed where the mapping constructed during the first step is modified so that the other two boundaries will also be mapped correctly.

In our example, we will first define curves that connect curves 1 and 2 such that only curves 1 and 2 will be mapped correctly. Afterwards, we will modify the connecting curves so that curves 3 and 4 will also be mapped correctly.

Curves which connect curves 1 and 2 are described by the following Hermite interpolation expressions:

$$x'(\xi,\eta) = X_1(\xi)\, h_1(\eta) + X_2(\xi)\, h_2(\eta) + \frac{\partial x(\xi,\eta=0)}{\partial \eta}\, h_3(\eta) + \frac{\partial x(\xi,\eta=1)}{\partial \eta}\, h_4(\eta) \tag{2.21a}$$

$$y'(\xi,\eta) = Y_1(\xi)\, h_1(\eta) + Y_2(\xi)\, h_2(\eta) + \frac{\partial y(\xi,\eta=0)}{\partial \eta}\, h_3(\eta) + \frac{\partial y(\xi,\eta=1)}{\partial \eta}\, h_4(\eta) \tag{2.21b}$$

Here, h_1, h_2, h_3, and h_4 are given by equation (2.11). The values of the partial derivatives in equation (2.21) will be given shortly.

By using equation (2.21), we map curves 1 and 2 in the x-y-t coordinate system to coordinate lines $\eta=0$ and $\eta=1$ in the ξ-η-τ coordinate system (fig. 2-7). Note, however, that curves 3 and 4 have not been mapped to coordinate lines $\xi=0$ and $\xi=1$. Instead, connecting curves 3′ and 4′ have been generated in the spatial domain, and it is these curves which have been mapped to $\xi=0$ and $\xi=1$ in the transformed domain. In order to map curves 3 and 4 in the x-y-t coordinate system to coordinate lines $\xi=0$ and $\xi=1$ in the ξ-η-τ coordinate system, we must adjust curves 3′ and 4′ so that they coincide with curves 3 and 4. With this in mind, we define two quantities — Δx and Δy — such that

$$x(\xi,\eta) = x'(\xi,\eta) + \Delta x(\xi,\eta) \tag{2.22a}$$

$$y(\xi,\eta) = y'(\xi,\eta) + \Delta y(\xi,\eta) \tag{2.22b}$$

will map curves 3 and 4 to $\xi=0$ and $\xi=1$, respectively. Here, x' and y' are given by equation (2.21).

Δx and Δy are given by the following Hermite interpolation expressions:

$$\begin{aligned}\Delta x(\xi,\eta) = {} & [x(\xi=0,\eta) - x'(\xi=0,\eta)]\, h_5(\xi) \\ & + [x(\xi=1,\eta) - x'(\xi=1,\eta)]\, h_6(\xi) \\ & + [\frac{\partial x(\xi=0,\eta)}{\partial \xi} - \frac{\partial x'(\xi=0,\eta)}{\partial \xi}]\, h_7(\xi) \\ & + [\frac{\partial x(\xi=1,\eta)}{\partial \xi} - \frac{\partial x'(\xi=1,\eta)}{\partial \xi}]\, h_8(\xi)\end{aligned} \tag{2.23a}$$

$$\begin{aligned}\Delta y(\xi,\eta) = {} & [y(\xi=0,\eta) - y'(\xi=0,\eta)]\, h_5(\xi) \\ & + [y(\xi=1,\eta) - y'(\xi=1,\eta)]\, h_6(\xi) \\ & + [\frac{\partial y(\xi=0,\eta)}{\partial \xi} - \frac{\partial y'(\xi=0,\eta)}{\partial \xi}]\, h_7(\xi) \\ & + [\frac{\partial y(\xi=1,\eta)}{\partial \xi} - \frac{\partial y'(\xi=1,\eta)}{\partial \xi}]\, h_8(\xi)\end{aligned} \tag{2.23b}$$

where

$$\frac{\partial x'(\xi=0,\eta)}{\partial \xi} = h_1(\eta)\,\frac{\partial x(\xi=0,\eta=0)}{\partial \xi} + h_2(\eta)\,\frac{\partial x(\xi=0,\eta=1)}{\partial \xi} + h_3(\eta)\frac{\partial^2 x(\xi=0,\eta=0)}{\partial \xi \partial \eta} + h_4(\eta)\,\frac{\partial^2 x(\xi=0,\eta=1)}{\partial \xi \partial \eta} \tag{2.23c}$$

$$\frac{\partial x'(\xi=1,\eta)}{\partial \xi} = h_1(\eta)\,\frac{\partial x(\xi=1,\eta=0)}{\partial \xi} + h_2(\eta)\,\frac{\partial x(\xi=1,\eta=1)}{\partial \xi} + h_3(\eta)\frac{\partial^2 x(\xi=1,\eta=0)}{\partial \xi \partial \eta} + h_4(\eta)\,\frac{\partial^2 x(\xi=1,\eta=1)}{\partial \xi \partial \eta} \tag{2.23d}$$

$$\frac{\partial y'(\xi=0,\eta)}{\partial \xi} = h_1(\eta)\,\frac{\partial y(\xi=0,\eta=0)}{\partial \xi} + h_2(\eta)\,\frac{\partial y(\xi=0,\eta=1)}{\partial \xi} + h_3(\eta)\frac{\partial^2 y(\xi=0,\eta=0)}{\partial \xi \partial \eta} + h_4(\eta)\,\frac{\partial^2 y(\xi=0,\eta=1)}{\partial \xi \partial \eta} \tag{2.23e}$$

$$\frac{\partial y'(\xi=1,\eta)}{\partial \xi} = h_1(\eta)\,\frac{\partial y(\xi=1,\eta=0)}{\partial \xi} + h_2(\eta)\,\frac{\partial y(\xi=1,\eta=1)}{\partial \xi} + h_3(\eta)\frac{\partial^2 y(\xi=1,\eta=0)}{\partial \xi \partial \eta} + h_4(\eta)\,\frac{\partial^2 y(\xi=1,\eta=1)}{\partial \xi \partial \eta} \tag{2.23f}$$

$$h_5(\xi) = 2\xi^3 - 3\xi^2 + 1 \tag{2.24a}$$

$$h_6(\xi) = -2\xi^3 + 3\xi^2 \tag{2.24b}$$

$$h_7(\xi) = \xi^3 - 2\xi^2 + \xi \tag{2.24c}$$

$$h_8(\xi) = \xi^3 - \xi^2 \tag{2.24d}$$

If we substitute equations (2.23) and (2.24) into equation (2.22), then we obtain the desired expressions for $x(\xi,\eta)$ and $y(\xi,\eta)$ which describe the mapping between the spatial domain and the transformed domain.

We still need to specify the derivative terms in equations (2.23) and (2.24). Similar to the Two-Boundary Method, the first-order derivative terms are chosen so that the connecting curves will intersect the boundaries orthogonally. This time, however, the spatial domain is

two dimensional so that we must use the dot product instead of the cross product to specify orthogonality at a boundary. Connecting curves will intersect curve 1 orthogonally when the dot product of $\bar{e}_\xi$ (a vector tangent to curve 1) and $\bar{e}_\eta$ (the vector tangent to the connecting curve) is zero. It can be shown that this will be the case when

$$\frac{\partial x(\xi,\eta=0)}{\partial \eta} = -K_1(\xi)\frac{\partial Y_1(\xi)}{\partial \xi} \quad \text{and} \quad \frac{\partial y(\xi,\eta=0)}{\partial \eta} = K_1(\xi)\frac{\partial X_1(\xi)}{\partial \xi} \tag{2.25}$$

Following this line of reasoning, the expressions for the first-order derivative terms in equations (2.23) and (2.24) are given below:

$$\frac{\partial x(\xi,\eta=0)}{\partial \eta} = -K_1(\xi)\frac{\partial Y_1(\xi)}{\partial \xi} \;, \quad \frac{\partial y(\xi,\eta=0)}{\partial \eta} = K_1(\xi)\frac{\partial X_1(\xi)}{\partial \xi} \tag{2.26a}$$

$$\frac{\partial x(\xi,\eta=1)}{\partial \eta} = -K_2(\xi)\frac{\partial Y_2(\xi)}{\partial \xi} \;, \quad \frac{\partial y(\xi,\eta=1)}{\partial \eta} = K_2(\xi)\frac{\partial X_2(\xi)}{\partial \xi} \tag{2.26b}$$

$$\frac{\partial x(\xi=0,\eta)}{\partial \xi} = K_3(\eta)\frac{\partial Y_3(\eta)}{\partial \eta} \;, \quad \frac{\partial y(\xi=0,\eta)}{\partial \xi} = -K_3(\eta)\frac{\partial X_3(\eta)}{\partial \eta} \tag{2.26c}$$

$$\frac{\partial x(\xi=1,\eta)}{\partial \xi} = K_4(\eta)\frac{\partial Y_4(\eta)}{\partial \eta} \;, \quad \frac{\partial y(\xi=1,\eta)}{\partial \xi} = -K_4(\eta)\frac{\partial X_4(\eta)}{\partial \eta} \tag{2.26d}$$

For our example , $K_1(\xi)$ and $K_2(\xi)$ were chosen to be equal to 0.3, while $K_3(\eta)$ and $K_4(\eta)$ were chosen to be equal to 0.1.

Methods for determining the second-order derivative terms present in equation (2.24) are given in reference 40. In our example, these terms are all set equal to zero.

Step 6 - Discretize the Domain. We discretize the domain in the ξ-η-τ coordinate system by replacing the temporal domain with equally incremented time levels and by replacing the spatial domain with $IL \times JL$ equally spaced grid points (fig. 2-8(b)).

The time levels are described by equation (2.14). The grid points are located at (ξ_i,η_j) where

$$\xi_i = (i-1)\,\Delta\xi\ ,\quad i=1,\,2,\,\ldots,\,IL \tag{2.27a}$$

$$\eta_j = (j-1)\,\Delta\eta\ ,\quad j=1,\,2,\,\ldots,\,JL \tag{2.27b}$$

$$\Delta\xi = \frac{1}{IL-1} \qquad \Delta\eta = \frac{1}{JL-1} \tag{2.27c}$$

If we substitute equations (2.23), (2.24), (2.26), and (2.27) into equation (2.22), then we can obtain the locations of the grid points in the x-y-t coordinate system. Figure 2-8(a) shows the system of grid points for our problem in the x-y-t coordinate system obtained by using equation (2.22).

Step 7 - Control the Distribution of Grid Points. In this example, we choose not to use stretching functions to redistribute the grid points within the spatial domain. If redistribution is desired, then the procedure described in the previous example can be followed.

Step 8 - Calculate Metric Coefficients. For our 2-D, non-deforming spatial domain, the metric coefficients which need to be evaluated are τ_t, ξ_x, η_x, ξ_y, and η_y. These metric coefficients can be evaluated by using the following equations:

$$\tau_t = 1 \tag{2.28a}$$

$$\xi_x = y_\eta / J \tag{2.28b}$$

$$\eta_x = -y_\xi / J \tag{2.28c}$$

$$\xi_y = -x_\eta / J \tag{2.28d}$$

$$\eta_y = x_\xi / J \tag{2.28e}$$

where J is again the Jacobian but this time is given by

$$J = x_\xi y_\eta - x_\eta y_\xi \tag{2.28f}$$

As before, the partial derivative terms in equation (2.28) can be evaluated either analytically or numerically by using finite-difference formulas depending upon how the governing equations written in the boundary-fitted coordinate system are cast.

2.3 The Six-Boundary Method

The Six-Boundary Method for generating grid points is intended for 3-D spatial domains in which six boundaries of the spatial domain need to be mapped correctly from the spatial domain to the transformed domain. The Six-Boundary Method (refs. 39, 40, and 51) is an extension of the Two-Boundary Method and the Four-Boundary Method described in the previous two sections, and, as such, consists of the same eight major steps with minor variations, namely:

1. Define the nature of the coordinate transformation.
2. Select a time-stretching function.
3. Select the six boundaries of the spatial domain that must be mapped correctly.
4. Describe the six boundaries selected in Step 3 in parametric form.
5. Map the spatial domain to a transformed domain.
6. Discretize the domain (i.e., replace the continuous domain of the problem with time levels and grid points).
7. Control the distribution of the grid points with stretching functions.
8. Calculate the metric coefficients needed by the FD or the FV method to obtain solutions.

We shall describe each of the eight steps of the Six-Boundary Method by generating a grid system within the deforming, spatial domain shown in figure 2-9(a).

Step 1 - Define the Coordinate Transformation. For the 3-D, deforming spatial domain shown in figure 2-9(a), we seek a coordinate transformation of the form

$$(x,y,z,t) \leftrightarrow (\xi,\eta,\zeta,\tau) \tag{2-29a}$$

or, more specifically,

$$t = t(\tau) \tag{2-29b}$$
$$x = x(\xi,\eta,\zeta,\tau) \tag{2.29c}$$
$$y = y(\xi,\eta,\zeta,\tau) \tag{2.29d}$$
$$z = z(\xi,\eta,\zeta,\tau) \tag{2.29e}$$

Here, x, y, z, and t comprise the coordinate system of the spatial domain, and ξ, η, ζ, and τ make up the boundary-fitted coordinate system of the transformed domain (figs. 2-9(a) and 2-9(b)).

Step 2 - Select a Time-Stretching Function. A time-stretching function is not used and t is set equal to τ as shown by equation (2.2).

Step 3 - Select Six Boundaries of the Spatial Domain. Since the spatial domain of figure 2-9(a) has only six boundaries, all six boundaries are selected. We choose surfaces 1, 2, 3, 4, 5, and 6 to correspond to coordinate planes $\eta=0$, $\eta=1$, $\xi=0$, $\xi=1$, $\zeta=0$, and $\zeta=1$, respectively (fig. 2-9); that is,

$$X_1 = x(\xi, \eta=0,\zeta, \tau) = X_1(\xi,\zeta,\tau) \tag{2.30a}$$
$$Y_1 = y(\xi, \eta=0,\zeta,\tau) = Y_1(\xi,\zeta,\tau) \tag{2.30b}$$
$$X_2 = x(\xi, \eta=1,\zeta,\tau) = X_2(\xi,\zeta,\tau) \tag{2.30c}$$
$$Y_2 = y(\xi, \eta=1,\zeta,\tau) = Y_2(\xi,\zeta,\tau) \tag{2.30d}$$

$$X_3 = x(\xi=0, \eta,\zeta,\tau) = X_3(\eta,\zeta,\tau) \quad (2.30e)$$

$$Y_3 = y(\xi=0, \eta,\zeta,\tau) = Y_3(\eta,\zeta,\tau) \quad (2.30f)$$

$$X_4 = x(\xi=1, \eta,\zeta,\tau) = X_4(\eta,\zeta,\tau) \quad (2.30g)$$

$$Y_4 = y(\xi=1, \eta,\zeta,\tau) = Y_4(\eta,\zeta,\tau) \quad (2.30h)$$

$$X_5 = x(\xi, \eta,\zeta = 0,\tau) = X_5(\xi,\eta,\tau) \quad (2.30i)$$

$$Y_5 = y(\xi, \eta,\zeta = 0,\tau) = Y_5(\xi,\eta,\tau) \quad (2.30j)$$

$$X_6 = x(\xi, \eta,\zeta = 1,\tau) = X_6(\xi,\eta,\tau) \quad (2.30k)$$

$$Y_6 = y(\xi, \eta,\zeta = 1,\tau) = Y_6(\xi,\eta,\tau) \quad (2.30l)$$

Here, X_1 and Y_1 are the x- and y-coordinates of surface 1; X_2 and Y_2 are the x- and y-coordinates of surface 2; X_3 and Y_3 are the x- and y-coordinates of surface 3; X_4 and Y_4 are the x- and y-coordinates of surface 4; X_5 and Y_5 are the x- and y-coordinates of surface 5; X_6 and Y_6 are the x- and y-coordinates of surface 6.

Step 4 - Describe the Four Boundaries Selected in Parametric Form. Having selected six boundaries, we now represent these boundaries in parametric form. For this problem, information about the six boundary surfaces is given in the form of a set of discrete points which lie along the surfaces. Thus, interpolation techniques must be used to generate the parametric equations of the boundary surfaces. In GRID2D/3D, a new technique referred to as 3-D bidirectional Hermite Interpolation can be used (Section 3.0).

Step 5 - Map the Spatial Domain. We map the spatial domain to a transformed domain in three steps. In the first step, correctly map two of the six boundaries by using the Two-Boundary Method. In the second step, correct the mapping completed in the first step by ensuring two more boundaries are mapped correctly (same as Step 5 of the Four-Boundary Method). Finally, in the third step, correct the mapping completed in the second step by ensuring the remaining two boundaries are mapped correctly.

Surfaces 1 and 2 will be mapped correctly by the following Hermite interpolation expressions:

$$x'(\xi,\eta,\zeta,\tau) = X_1(\xi,\zeta,\tau)\, h_1(\eta) + X_2(\xi,\zeta,\tau)\, h_2(\eta)$$

$$+ \frac{\partial x(\xi,\eta=0,\zeta,\tau)}{\partial \eta}\, h_3(\eta) + \frac{\partial x(\xi,\eta=1,\zeta,\tau)}{\partial \eta}\, h_4(\eta) \tag{2.31a}$$

$$y'(\xi,\eta,\zeta,\tau) = Y_1(\xi,\zeta,\tau)\, h_1(\eta) + Y_2(\xi,\zeta,\tau)\, h_2(\eta)$$

$$+ \frac{\partial y(\xi,\eta=0,\zeta,\tau)}{\partial \eta}\, h_3(\eta) + \frac{\partial y(\xi,\eta=1,\zeta,\tau)}{\partial \eta}\, h_4(\eta) \tag{2.31b}$$

Here, h_1, h_2, h_3, and h_4 are given by equation (2.11).

Surfaces 1, 2, 3, and 4 will be mapped correctly by the following equations:

$$x''(\xi,\eta,\zeta,\tau) = x'(\xi,\eta,\zeta,\tau) + \Delta x'(\xi,\eta,\zeta,\tau) \tag{2.32a}$$

$$y''(\xi,\eta,\zeta,\tau) = y'(\xi,\eta,\zeta,\tau) + \Delta y'(\xi,\eta,\zeta,\tau) \tag{2.32b}$$

where x' and y' are given by equation (2-31) and

$$\Delta x'(\xi,\eta,\zeta,\tau) = [X_3(\eta,\zeta,\tau) - x'(\xi=0,\eta,\zeta,\tau)]\, h_5(\xi)$$

$$+ [X_4(\eta,\zeta,\tau) - x'(\xi=1,\eta,\zeta,\tau)]\, h_6(\xi)$$

$$+ [\frac{\partial x(\xi=0,\eta,\zeta,\tau)}{\partial \xi} - \frac{\partial x'(\xi=0,\eta,\zeta,\tau)}{\partial \xi}]\, h_7(\xi)$$

$$+ [\frac{\partial x(\xi=1,\eta,\zeta,\tau)}{\partial \xi} - \frac{\partial x'(\xi=1,\eta,\zeta,\tau)}{\partial \xi}]\, h_8(\xi) \tag{2.33a}$$

$$\Delta y'(\xi,\eta,\zeta,\tau) = [Y_3(\eta,\zeta,\tau) - y'(\xi=0,\eta,\zeta,\tau)]\, h_5(\xi)$$

$$+ [Y_4(\eta,\zeta,\tau) - y'(\xi=1,\eta,\zeta,\tau)]\, h_6(\xi)$$

$$+ [\frac{\partial y(\xi=0,\eta,\zeta,\tau)}{\partial \xi} - \frac{\partial y'(\xi=0,\eta,\zeta,\tau)}{\partial \xi}]\, h_7(\xi)$$

$$+ [\frac{\partial y(\xi=1,\eta,\zeta,\tau)}{\partial \xi} - \frac{\partial y'(\xi=1,\eta,\zeta,\tau)}{\partial \xi}]\, h_8(\xi) \tag{2.33b}$$

In the above equations, h_5, h_6, h_7, and h_8 are given by equation (2.24).

All six surfaces of the spatial domain shown in figure 2-9(a) – namely, surfaces 1, 2, 3, 4, 5, and 6 – will be mapped correctly by the following equations:

$$x(\xi,\eta,\zeta,\tau) = x''(\xi,\eta,\zeta,\tau) + \Delta x''(\xi,\eta,\zeta,\tau) \tag{2.34a}$$

$$y(\xi,\eta,\zeta,\tau) = y''(\xi,\eta,\zeta,\tau) + \Delta y''(\xi,\eta,\zeta,\tau) \tag{2.34b}$$

where x'' and y'' are given by equation (2-32) and

$$\begin{aligned}\Delta x''(\xi,\eta,\zeta,\tau) = {} & [X_5(\xi,\eta,\tau) - x''(\xi,\eta,\zeta=0,\tau)]\, h_9(\xi) \\ & + [X_6(\xi,\eta,\tau) - x''(\xi,\eta,\zeta=1,\tau)]\, h_{10}(\xi) \\ & + \left[\frac{\partial x(\xi,\eta,\zeta=0,\tau)}{\partial \xi} - \frac{\partial x''(\xi,\eta,\zeta=0,\tau)}{\partial \xi}\right] h_{11}(\xi) \\ & + \left[\frac{\partial x(\xi,\eta,\zeta=1,\tau)}{\partial \xi} - \frac{\partial x''(\xi,\eta,\zeta=1,\tau)}{\partial \xi}\right] h_{12}(\xi)\end{aligned} \tag{2.35a}$$

$$\begin{aligned}\Delta y'(\xi,\eta,\zeta,\tau) = {} & [Y_3(\eta,\zeta,\tau) - y'(\xi=0,\eta,\zeta,\tau)]\, h_5(\xi) \\ & + [Y_4(\eta,\zeta,\tau) - y'(\xi=1,\eta,\zeta,\tau)]\, h_6(\xi) \\ & + \left[\frac{\partial y(\xi=0,\eta,\zeta,\tau)}{\partial \xi} - \frac{\partial y'(\xi=0,\eta,\zeta,\tau)}{\partial \xi}\right] h_7(\xi) \\ & + \left[\frac{\partial y(\xi=1,\eta,\zeta,\tau)}{\partial \xi} - \frac{\partial y'(\xi=1,\eta,\zeta,\tau)}{\partial \xi}\right] h_8(\xi)\end{aligned} \tag{2.35b}$$

$$h_9(\zeta) = 2\zeta^3 - 3\zeta^2 + 1 \tag{2.36a}$$

$$h_{10}(\zeta) = -2\zeta^3 + 3\zeta^2 \tag{2.36b}$$

$$h_{11}(\zeta) = \zeta^3 - 2\zeta^2 + \zeta \tag{2.36c}$$

$$h_{12}(\zeta) = \zeta^3 - \zeta^2 \tag{2.36d}$$

If we substitute equations (2.31) to (2.33) into equation (2.34), then we obtain the desired expressions for $x(\xi,\eta,\zeta,\tau)$ and $y(\xi,\eta,\zeta,\tau)$ which describe the mapping between the spatial domain and the transformed domain.

We still need to specify the derivative terms in equations (2.31), (2.33), and (2.35). Similar to the Two- and Four-Boundary Methods, the first-order derivative terms are chosen so that the connecting curves will intersect the boundaries orthogonally. Methods for determining the second-order derivative terms are given in reference 40. In our example, these terms are all set equal to zero.

Step 6 - Discretize the Domain. We discretize the domain in the ξ-η-ζ-τ coordinate system by replacing the temporal domain with equally incremented time levels and by replacing the spatial domain with $IL \times JL \times KL$ equally spaced grid points.

The time levels are described by equation (2.14). The grid points are located at (ξ_i,η_j,ζ_k) where ξ_i, η_j, ζ_k are given by equation (2.15).

If we substitute equation (2.14) and (2.15) into equation (2.34), we can obtain the locations of the grid points in the x-y-t coordinate system.

Step 7 - Control the Distribution of Grid Points. In this example, we choose not to use stretching functions to redistribute the grid points within the spatial domain. If redistribution is desired, then the procedure described in the example in Section 2.1 can be followed.

Step 8 - Calculate Metric Coefficients. For our 3-D, deforming spatial domain, the metric coefficients which need to be evaluated are given by equation (2.18).

2.4 Additional Remarks

We conclude this section by mentioning three problems which must be dealt with when using the Two-, Four, and Six-Boundary Methods.

First, slope discontinuities present in the boundaries of spatial domains will propagate into the interior of the grid systems generated by using these methods. These discontinuities

in the slopes of the grid lines are undesirable since they can lead to errors in the solution. To correct for this, some technique should be used to smooth the grid. One way to smooth a grid system with slope discontinuity is to apply a Laplacian operator to the region near the discontinuity. This method is described in Section 4.0.

Second, care must be taken when choosing the "K factors" and the stretching functions for the Two-, Four-, and Six-Boundary Methods. Large "K factors" tend to produce grid lines with more curvature. Such grid lines often overlap in the spatial domain or, at least, can form a grid system which is very skewed. In general, numerical values in the "K factors" and the stretching functions are arrived at in an iterative manner. First, a grid is generated by using one set of inputs for the "K factors" and the stretching functions. Next, that grid is plotted (GRID2D/3D contains a graphics program to plot grid systems that it generates) and inspected visually. Based on that inspection, the inputs are modified accordingly. This process repeats until a satisfactory grid has been obtained. Since GRID2D/3D is highly efficient, an acceptable grid system can be generated within a short time.

Last, when connecting curves are discretized to form grid points, the orthogonality which was forced at the boundaries may be lost. Figure 2-10 illustrates this point. Between grid points a and b, curve c is approximated by line segment a-b. The original 90^o angle, α, between boundary d and curve c has been replaced by angle β between boundary d and line segment a-b. In order that β more nearly approximate α, two things can be done. First, stretching functions can be used to move point a closer to point b. Second, larger "K factors" can be utilized to force the effect of orthogonality further into the domain along curve c. In this way, orthogonality between the grid lines and the boundary curves can be maintained after the discretization of the spatial domain.

3.0 METHODS FOR GENERATING PARAMETRIC REPRESENTATION OF BOUNDARIES

In Section 2.0, it was shown that in order to use the Two-, Four-, and Six-Boundary Methods, it is necessary to represent the boundaries of the spatial domain in parametric form. These boundaries are curves for two-dimensional spatial domains and surfaces for three-dimensional ones. In this section, methods for representing curves and surfaces in parametric form are described.

3.1 Parametric Representation of Curves and Surfaces

Curves and surfaces can be described mathematically in several different ways. For example, a curve in the xy-plane can be represented by

$$y = f(x) \tag{3.1}$$

or

$$f(x,y) = 0 \tag{3.2}$$

Alternatively, we can describe the same curve by

$$x = g(s) \quad y = f[g(s)] = h(s) \tag{3.3}$$

Equation (3.3) is a parametric representation of the curve in terms of a parameter, s. The choice of s is rather arbitrary, the only restriction being that s must increase monotonically along the curve. Here, we note that all curves and surfaces can be represented by parametric equations and that there is no one unique way of representing a curve or surface in parametric form.

In grid generation, information about a curve or surface is given either by an analytical expression such as equation (3.1) or by a set of coordinates which describe the locations of a finite number of discrete points on the curve or surface. When information about a curve or surface is provided in the form of an analytical expression (such as eq. (3.1)), it is a straightforward matter to generate a set of parametric equations (such as eq. (3.3)) for the curve or surface. When information about a curve or surface is given by a finite number of discrete points, then some type of interpolation schemes must first be used to approximate the curve or surface by an analytical expression before it can be represented in parametric form. The following subsections present methods to obtain parametric equations for curves and surfaces given as sets of discrete points.

3.2 Approximation of Curves in Two and Three Dimensions

For 2-D spatial domains, Lagrange interpolation is usually unsatisfactory for representing curves because it produces curves which may oscillate wildly when the number of discrete points (henceforth referred to as nodal points) is large (e.g., ref. 55). Hermite interpolation is usually impractical as well, since it requires information about derivative values at the nodal points which is seldom available (e.g., ref. 56). Least-squares regression is not computationally efficient for large sets of nodal points and yields curves which do not, in general, pass through each nodal point (e.g., ref. 57). Spline interpolation, on the other hand, yields curves that do pass through each nodal point and are well behaved (i.e., they do not oscillate wildly). Spline interpolation uses a different function (typically a low-degree polynomial) between successive nodal points (fig. 3-1). The resulting piecewise curve is made smooth by enforcing continuity of as many derivatives as possible at the nodal points. This technique can easily be used to approximate curves which are functions of two or three spatial coordinates. Because of the aforementioned attractive features of spline interpolation, this

method will be used here to approximate curves in both two and three dimensions. We now consider two types of splines – cubic splines and tension splines.

3.2.1 Cubic Spline Interpolation. – The objective in cubic spline interpolation is to derive a different third-degree polynomial for each interval between successive nodal points, subject to the condition that the polynomials be connected piecewise to form a smooth curve that is continuous up to the second-order derivative. Since we require the curve to be written in parametric form, the equation describing this curve in an arbitrary interval, i, can be written as (fig. 3-1)

$$X_i(s) = A_i s^3 + B_i s^2 + C_i s + D_i \tag{3.4}$$

Here, X represents any spatial coordinate (e.g., x, y, or z in the Cartesian system), and s is a parameter which was mentioned previously and which will be discussed in more depth later in this subsection. A_i, B_i, C_i, and D_i are coefficients which vary from interval to interval. For $n+1$ nodal points, there are n intervals and $4n$ such coefficients; thus, $4n$ conditions are required in order to evaluate these coefficients. These conditions are summarized below:

1. The spline curve must pass through the nodal points. This gives $2n$ conditions.
2. The first- and second-order derivatives of the spline curve must be continuous at all interior nodal points (i.e., nodal points 1 through $n-1$; fig. 3-1). This gives $2n-2$ conditions.
3. Two conditions control the behavior of the spline curve at the two end nodal points. These conditions will be discussed later in this subsection.

We could use these $4n$ conditions together with equation (3.4) to obtain $4n$ simultaneous equations in $4n$ unknowns. However, a shortcut is possible which requires the

solution of only $n-1$ simultaneous equations (e.g., refs. 55-59). Since the shortcut method is much more efficient, it is used in GRID2D/3D.

We begin the shortcut method by noting that since the curve for each interval is a cubic, the second-order derivative within each interval is a straight line. Thus, we can write

$$X_i''(s) = X''(s_{i-1})\frac{(s-s_i)}{(s_{i-1}-s_i)} + X''(s_i)\frac{(s-s_{i-1})}{(s_i-s_{i-1})} \tag{3.5}$$

where $X_i''(s)$ is the value of the second-order derivative of X at any location s within the i^{th} interval (fig. 3-1). The parameter s takes on the values s_{i-1} and s_i at the beginning and end of the interval, respectively.

By integrating equation (3.5) twice with respect to s, we obtain an expression for $X_i(s)$ which contains two constants of integration. We solve for these constants by imposing the conditions that $X_i(s)$ must equal $X(s_{i-1})$ at s_{i-1} and $X_i(s)$ must equal $X(s_i)$ at s_i. This yields

$$\begin{aligned} X_i(s) = {} & \frac{X''(s_{i-1})}{6(s_i-s_{i-1})}(s_i-s)^3 + \frac{X''(s_i)}{6(s_i-s_{i-1})}(s-s_{i-1})^3 \\ & + \left(\frac{X(s_{i-1})}{(s_i-s_{i-1})} - \frac{X''(s_{i-1})(s_i-s_{i-1})}{6}\right)(s_i-s) \\ & + \left(\frac{X(s_i)}{(s_i-s_{i-1})} - \frac{X''(s_i)(s_i-s_{i-1})}{6}\right)(s-s_{i-1}) \end{aligned} \tag{3.6}$$

The only unknowns in the above equation are the second-order derivatives at the beginning and end of the interval – $X''(s_{i-1})$ and $X''(s_i)$. Values of these second-order derivative terms can be found by invoking the condition that the first derivatives at the nodal points must be continuous:

$$X_{i-1}'(s_i) = X_i'(s_i) \tag{3.7}$$

Equation (3.6) can be differentiated to give an expression for $X_i'(s)$. If we do this for both the $(i-1)^{th}$ and the i^{th} intervals and set the two results equal to each other according to equation (3.7), we obtain the following result:

$$(s_i-s_{i-1})X''(s_{i-1}) + 2(s_{i+1}-s_{i-1})X''(s_i) + (s_{i+1}-s_i)X''(s_{i+1})$$

$$= \frac{6}{(s_{i+1}-s_i)}[X(s_{i+1})-X(s_i)] - \frac{6}{(s_i-s_{i-1})}[X(s_i)-X(s_{i-1})] \tag{3.8}$$

If equation (3.8) is applied at all interior nodal points, then we obtain $n-1$ simultaneous equations in $n+1$ unknown second-order derivative terms. This system of simultaneous equations has a tridiagonal coefficient matrix and requires two more conditions before it can be solved. Here is where our two additional conditions regarding the end nodal points are necessary. There are several conditions which can be imposed at the end nodal points. We mention only the following two:

1. Cyclic End Conditions. If the curve being approximated is cyclic (i.e., the two end nodal points represent the same point in space as in a closed curve), then it is appropriate to write

   $$X'(s_0) = X'(s_n) \quad \text{and} \quad X''(s_0) = X''(s_n) \tag{3.9}$$

   With this assumption, the coefficient matrix of the system of equations mentioned previously is now cyclic tridiagonal, and the system becomes $n+1$ equations in $n+1$ unknowns. An example of a cubic spline with cyclic end conditions is shown in figure 3-2(a).

2. Natural End Conditions. If we allow the second-order derivative of the spline curve to be zero at the two end nodal points, then we minimize the total curvature of the spline curve. This is the so-called natural spline, and the end conditions are given by

$$X''(s_0) = X''(s_n) = 0 \tag{3.10}$$

In this case, the system of equations mentioned previously reduces to $n-1$ equations in $n-1$ unknowns. An example of a cubic spline with natural end conditions is shown in figure 3-2(b).

In either case, the system of equations in $X''(s_i)$ can be solved efficiently using simple computational algorithms (e.g., refs. 55 to 59). Once the $X''(s_i)$'s are known, values for X at any location s along the curve can be found using equation (3.6). Recall that X in equation (3.6) represents any spatial coordinate – x, y or z (comments under equation (3.4)). Thus, the required parametric equation is given by $x = X(s)$, $y = Y(s)$ for plane curves and $x = X(s)$, $y = Y(s)$, $z = Z(s)$ for twisted curves and surfaces.

We now return to the question of how to choose the parameter, s. One simple and effective way is to let s represent arc length along the curve. Of course, we do not know the actual arc length a priori, but an adequate estimate can be obtained by summing linear distances between nodal points. Thus, for a two-dimensional curve in the xy-plane (i.e., a curve which is a function of only two spatial coordinates -- say x and y), we have $s_0 = 0$, $s_1 = s_0 + \sqrt{[(X_2-X_1)^2+(Y_2-Y_1)^2]}$, $s_2 = s_1 + \sqrt{[(X_3-X_2)^2+(Y_3-Y_2)^2]}$, and so on.

3.2.2 Tension Spline Interpolation. – Cubic spline interpolation produces curves that are well behaved in most cases; however, in cases where the curve to be approximated possesses extreme curvature, curves generated using cubic spline interpolation often have unwanted wiggles. This phenomenon is illustrated in figure 3-3. One solution to this problem is to put the spline in tension. One might visualize this as pulling on the ends of the cubic spline to "straighten out" any unwanted wiggles. We add tension to the cubic spline by replacing equation (3.5) by

$$X_i''(s) - \sigma^2 X_i(s) = [X''(s_{i-1}) - \sigma^2 X(s_{i-1})]\frac{(s_i - s)}{(s_i - s_{i-1})} + [X''(s_i) - \sigma^2 X(s_i)]\frac{(s - s_{i-1})}{(s_i - s_{i-1})} \tag{3.11}$$

where σ is the tension parameter, the range of which is given by $0 < \sigma < \infty$. The spline curve created by adding tension to the cubic spline is known as a tension spline (ref. 56). The tension spline tends toward a linear spline as σ is increased and approaches a cubic spline for values of σ near zero.

As with the cubic spline, we integrate equation (3.11) twice with respect to s and require that the curve pass through the proper nodal points to give

$$X_i(s) = \frac{X''(s_{i-1})}{\sigma^2}\frac{sinh[\sigma(s_i - s)]}{sinh[\sigma(s_i - s_{i-1})]} + [X(s_{i-1}) - \frac{X''(s_{i-1})}{\sigma^2}]\frac{(s_i - s)}{(s_i - s_{i-1})} + \frac{X''(s_i)}{\sigma^2}\frac{sinh[\sigma(s - s_{i-1})]}{sinh[\sigma(s_i - s_{i-1})]} + [X(s_i) - \frac{X''(s_i)}{\sigma^2}]\frac{(s - s_{i-1})}{(s_i - s_{i-1})} \tag{3.12}$$

Upon demanding continuity of the first derivative at the interior nodal points, we have

$$\begin{aligned}
&\left\{\frac{1}{(s_i - s_{i-1})} - \frac{\sigma}{sinh[\sigma(s_i - s_{i-1})]}\right\}\frac{X''(s_{i-1})}{\sigma^2} \\
&\quad + \left\{\frac{\sigma cosh[\sigma(s_i - s_{i-1})]}{sinh[\sigma(s_i - s_{i-1})]} - \frac{1}{(s_i - s_{i-1})}\right. \\
&\quad \left. + \frac{\sigma cosh[\sigma(s_{i+1} - s_i)]}{sinh[\sigma(s_{i+1} - s_i)]} - \frac{1}{(s_{i+1} - s_i)}\right\}\frac{X''(s_i)}{\sigma^2} \\
&\quad + \left\{\frac{1}{(s_{i+1} - s_i)} - \frac{\sigma}{sinh[\sigma(s_{i+1} - s_i)]}\right\}\frac{X''(s_{i+1})}{\sigma^2} \\
&= \left(\frac{X(s_{i+1}) - X(s_i)}{(s_{i+1} - s_i)} - \frac{X(s_i) - X(s_{i-1})}{(s_i - s_{i-1})}\right)
\end{aligned} \tag{3.13}$$

Applying equation (3.13) at all interior nodal points again results in a system of $n-1$ equations in $n+1$ unknowns with a tridiagonal coefficient matrix. Two conditions are required at the end nodal points to make solution of the system possible. The cyclic and natural end

conditions presented for the cubic spline can be used for the tension spline as well with $X''(s_i)$ replaced by $X''(s_i) - \sigma^2 X(s_i)$. In either case, the resulting system of simultaneous linear equations can be solved by using simple numerical algorithms.

Since tension splines involve hyperbolic functions, they are computationally less efficient than cubic splines; however, excellent results can be obtained for cases where cubic splines prove unsatisfactory. Figure 3-4 shows that a tension spline with $\sigma=10$ exhibits none of the wiggles visually present in the cubic spline of figure 3-3. The same set of nodal points was used to generate the spline curves shown in figures 3-3 and 3-4.

3.3 Approximation of Surfaces in Three Dimensions

For three-dimensional spatial domains, the boundaries of the domain are surfaces. As one might expect, methods for describing surfaces based on interpolating between specified nodal points which lie on the surface are more complex than their two-dimensional counterparts. Part of the additional difficulty lies in the fact that there are several ways in which information about discrete points on a surface may be given. With curves, it is sufficient to start at one end of the curve and specify nodal points at random intervals as one moves along the length of the curve. This is not the case for surfaces. One cannot just randomly pick points on a surface and expect to efficiently construct an approximation of the surface using an interpolation method unless the points are chosen in an organized manner. Here, we consider the following two structured methods for specifying points on a surface:

Case 1. Only points which lie along the edges of the surface are given. These edges exist as twisted (3-D) or plane (2-D) curves. This case is applicable to surfaces whose interior regions are similar to their edge contours. Examples of such surfaces are shown in figure 3-5.

Case 2. Points on the surface are given on a grid as shown in figure 3-6. In figure 3-6, the z-coordinates of surface H are given on a rectangular grid in the Cartesian system. This case is applicable to any surface but is especially well-suited to surfaces whose interior regions differ greatly from their edge contours. Examples of such surfaces are shown in figure 3-7.

In the next three subsections, we present three methods for approximating boundary surfaces by interpolating between points which lie on the surfaces. Section 3.3.1 describes a method, referred to as transfinite interpolation with bilinear blending (also known as linear Coon's interpolation), which can be used when the nodal points are given in the format of Case 1 above. Section 3.3.2 presents a new method, referred to as three-dimensional bidirectional Hermite interpolation, which can also be used when the nodal points are given in the format of Case 1. Section 3.3.3 presents a method, referred to as parametric bihyperbolic spline interpolation, which can be used when the nodal points are given in the format of Case 2 above.

3.3.1 Transfinite Interpolation With Bilinear Blending – To illustrate how transfinite interpolation with bilinear blending approximates surfaces, consider the boundary surface shown in figure 3-8. That boundary surface is bounded by four twisted curves. Information about the four twisted curves may be given in analytical form, or they may be put into analytical form from coordinates of the nodal points located on the curves by using either cubic spline or tension spline interpolation described in Section 3.2. In either case, we end up with parametric equations describing each of the curves in the following form:

$$\begin{array}{llll} X_1=X_1(s_1) & X_2=X_2(s_2) & X_3=X_3(s_3) & X_4=X_4(s_4) \\ Y_1=Y_1(s_1) & Y_2=Y_2(s_2) & Y_3=Y_3(s_3) & Y_4=Y_4(s_4) \\ Z_1=Z_1(s_1) & Z_2=Z_2(s_2) & Z_3=Z_3(s_3) & Z_4=Z_4(s_4) \end{array}$$

Here, X_i, Y_i, and Z_i describe curve i where i = 1, 2, 3, 4, and s_i represents the approximate arc length along curve i as discussed in Section 3.3.1.

Recall that for the Two-, Four- and Six-Boundary Methods, boundary surfaces in the spatial domain are mapped onto coordinate planes in the transformed domain. Here, we map the boundary surface shown in figure 3-8 onto the coordinate plane shown in figure 3-9 located at $\zeta = 0$ in the transformed domain. Accordingly, we relate s_1, s_2, s_3, and s_4 to ξ and η as follows:

$$s_1 = s_1(\xi,\eta{=}0,\zeta{=}0) = s_1(\xi) \tag{3.14a}$$
$$s_2 = s_2(\xi,\eta{=}1,\zeta{=}0) = s_2(\xi) \tag{3.14b}$$
$$s_3 = s_3(\xi{=}0,\eta,\ \zeta{=}0) = s_3(\eta) \tag{3.14c}$$
$$s_4 = s_4(\xi{=}1,\eta,\zeta{=}0) = s_4(\eta) \tag{3.14d}$$

By using equation (3.14) above, we linearly interpolate between curves 1 and 2 to obtain the following equations:

$$x_{12}(\xi,\eta,\zeta{=}0) = (1{-}\eta)\ X_1(\xi) + \eta\ X_2(\xi) \tag{3.15a}$$
$$y_{12}(\xi,\eta,\zeta{=}0) = (1{-}\eta)\ Y_1(\xi) + \eta\ Y_2(\xi) \tag{3.15b}$$
$$z_{12}(\xi,\eta,\zeta{=}0) = (1{-}\eta)\ Z_1(\xi) + \eta\ Z_2(\xi) \tag{3.15c}$$

The surface described by the above equations is known as a lofted surface between curves 1 and 2 and is shown in figure 3-10(a).

In a similar manner, we linearly interpolate between curves 3 and 4 to form a lofted surface described by

$$x_{34}(\xi,\eta,\zeta{=}0) = (1{-}\xi)\ X_3(\eta) + \xi\ X_4(\eta) \tag{3.16a}$$
$$y_{34}(\xi,\eta,\zeta{=}0) = (1{-}\xi)\ Y_3(\eta) + \xi\ Y_4(\eta) \tag{3.16b}$$
$$z_{34}(\xi,\eta,\zeta{=}0) = (1{-}\xi)\ Z_3(\eta) + \xi\ Z_4(\eta) \tag{3.16c}$$

This surface is shown in figure 3-10(b).

If we add these two surfaces, we obtain a surface which does not pass through the four corners determined by the intersections of the four twisted curves. In order to correct for this, we define a third surface by

$$x_{1234}(\xi,\eta,\zeta=0) = (1-\xi)(1-\eta)\, X_1(\xi=0) + \xi\,(1-\eta)\, X_1(\xi=1) + (1-\xi)\,\eta\, X_2(\xi=0) + \xi\,\eta\, X_2(\xi=1) \tag{3.17a}$$

$$y_{1234}(\xi,\eta,\zeta=0) = (1-\xi)(1-\eta)\, Y_1(\xi=0) + \xi\,(1-\eta)\, Y_1(\xi=1) + (1-\xi)\,\eta\, Y_2(\xi=0) + \xi\,\eta\, Y_2(\xi=1) \tag{3.17b}$$

$$z_{1234}(\xi,\eta,\zeta=0) = (1-\xi)(1-\eta)\, Z_1(\xi=0) + \xi\,(1-\eta)\, Z_1(\xi=1) + (1-\xi)\,\eta\, Z_2(\xi=0) + \xi\,\eta\, Z_2(\xi=1) \tag{3.17c}$$

This surface interpolates the four corners of the region and is simply the plane section shown in figure 3-10(c).

By adding equations (3.15) and (3.16) and subtracting equation (3.17) from the result, we obtain the following expressions which describe the surface bounded by curves 1 through 4 (fig. 3-10(d)):

$$x(\xi,\eta,\zeta=0) = (1-\eta)\, X_1(\xi) + \eta\, X_2(\xi) + (1-\xi)\, X_3(\eta) + \xi\, X_4(\eta) - [(1-\xi)(1-\eta)\, X_1(\xi=0) + \xi\,(1-\eta)\, X_1(\xi=1) + (1-\xi)\,\eta\, X_2(\xi=0) + \xi\,\eta\, X_2(\xi=1)] \tag{3.18a}$$

$$y(\xi,\eta,\zeta=0) = (1-\eta)\, Y_1(\xi) + \eta\, Y_2(\xi) + (1-\xi)\, Y_3(\eta) + \xi\, Y_4(\eta) - [(1-\xi)(1-\eta)\, Y_1(\xi=0) + \xi\,(1-\eta)\, Y_1(\xi=1) + (1-\xi)\,\eta\, Y_2(\xi=0) + \xi\,\eta\, Y_2(\xi=1)] \tag{3.18b}$$

$$z(\xi,\eta,\zeta=0) = (1-\eta)\, Z_1(\xi) + \eta\, Z_2(\xi) + (1-\xi)\, Z_3(\eta) + \xi\, Z_4(\eta) - [(1-\xi)(1-\eta)\, Z_1(\xi=0) + \xi\,(1-\eta)\, Z_1(\xi=1) + (1-\xi)\,\eta\, Z_2(\xi=0) + \xi\,\eta\, Z_2(\xi=1)] \tag{3.18c}$$

Approximating a surface, by linear interpolation between curves that bound it, is known as transfinite bilinear interpolation. This method was named by Gordon and Hall (ref. 34), but was first demonstrated by Coons (ref. 33). Thus, this method is also referred to as linear Coon's interpolation.

For some surfaces, transfinite bilinear interpolation does not produce a satisfactory approximation. This usually occurs when the curves that bound the surface possess extreme curvatures. In such cases, higher degree blending functions (e.g., cubic Hermite blending functions) can be employed and this is the subject of the next section. Also, if additional information about contour lines on the interior of the surface is available, one can break the surface up into simpler subsurfaces and approximate each subsurface separately.

3.3.2 Three-Dimensional Bidirectional Hermite Interpolation. — As noted at the conclusion of the previous section, transfinite bilinear interpolation sometimes does not produce satisfactory surface approximations. Also, if a boundary surface is broken up into two or more subsurfaces with each subsurface generated by transfinite bilinear interpolation, then that boundary surface will have discontinuous first-order derivatives at all interfaces where different subsurfaces connect. Here, a new technique for generating surfaces, referred to as 3-D bidirectional Hermite interpolation, has been developed which does not have the aforementioned shortcomings of the transfinite bilinear interpolation.

To illustrate the 3-D bidirectional Hermite interpolation, again consider the boundary surface bounded by four twisted curves shown in figure 3-8. Information about the four twisted curves may be given in analytical form, or they may be put into analytical form from coordinates of the nodal points located on the curves by using either cubic spline or tension spline interpolation described in Section 3.2. In either case, we end up with parametric equations describing each of the curves in the following form:

$$X_i = X_i(s_i) \quad Y_i = Y_i(s_i) \quad Z_i = Z_i(s_i) \tag{3.19}$$

where X_i, Y_i, and Z_i describe curve i and $i = 1, 2, 3, 4$. The parameter, s_i, represents the approximate arc length along curve i as discussed in Section 3.3.1.

Recall that for the Two-, Four-, and Six-Boundary Methods, boundary surfaces in the spatial domain are mapped onto coordinate planes in the transformed domain. Here, we map the boundary surface shown in figure 3-8 onto the coordinate plane shown in figure 3-9 located at $\zeta = 0$ in the transformed domain. Accordingly, we relate the s_i parameters to ξ and η by equation (3.14).

By using equation (3.14) and the Four-Boundary Method with Hermite interpolants described in Section 2.2, we obtain

$$x(\xi,\eta,\zeta=0) = x'_{12}(\xi,\eta,\zeta=0) + \Delta x(\xi,\eta,\zeta=0) \tag{3.20a}$$
$$y(\xi,\eta,\zeta=0) = y'_{12}(\xi,\eta,\zeta=0) + \Delta y(\xi,\eta,\zeta=0) \tag{3.20b}$$
$$z(\xi,\eta,\zeta=0) = z'_{12}(\xi,\eta,\zeta=0) + \Delta z(\xi,\eta,\zeta=0) \tag{3.20c}$$

where

$$x'_{12}(\xi,\eta,\zeta=0) = X_1(\xi)\, h_1(\eta) + X_2(\xi)\, h_2(\eta) + \frac{\partial x(\xi,\eta=0,\zeta=0)}{\partial \eta}\, h_3(\eta) + \frac{\partial x(\xi,\eta=1,\zeta=0)}{\partial \eta}\, h_4(\eta) \tag{3.20d}$$

$$y'_{12}(\xi,\eta,\zeta=0) = Y_1(\xi)\, h_1(\eta) + Y_2(\xi)\, h_2(\eta) + \frac{\partial y(\xi,\eta=0,\zeta=0)}{\partial \eta}\, h_3(\eta) + \frac{\partial y(\xi,\eta=1,\zeta=0)}{\partial \eta}\, h_4(\eta) \tag{3.20e}$$

$$z'_{12}(\xi,\eta,\zeta=0) = Z_1(\xi)\, h_1(\eta) + Z_2(\xi)\, h_2(\eta) + \frac{\partial z(\xi,\eta=0,\zeta=0)}{\partial \eta}\, h_3(\eta) + \frac{\partial z(\xi,\eta=1,\zeta=0)}{\partial \eta}\, h_4(\eta) \tag{3.20f}$$

$$\Delta x(\xi,\eta,\zeta=0) = [x(\xi=0,\eta,\zeta=0) - x'_{12}(\xi=0,\eta,\zeta=0)]\, h_5(\xi)$$
$$+ [x(\xi=1,\eta,\zeta=0) - x'_{12}(\xi=1,\eta,\zeta=0)]\, h_6(\xi)$$
$$+ [\frac{\partial x(\xi=0,\eta,\zeta=0)}{\partial \xi} - \frac{\partial x'_{12}(\xi=0,\eta,\zeta=0)}{\partial \xi}]\, h_7(\xi)$$
$$+ [\frac{\partial x(\xi=1,\eta,\zeta=0)}{\partial \xi} - \frac{\partial x'_{12}(\xi=1,\eta,\zeta=0)}{\partial \xi}]\, h_8(\xi) \qquad (3.20g)$$

$$\Delta y(\xi,\eta,\zeta=0) = [y(\xi=0,\eta,\zeta=0) - y'_{12}(\xi=0,\eta,\zeta=0)]\, h_5(\xi)$$
$$+ [y(\xi=1,\eta,\zeta=0) - y'_{12}(\xi=1,\eta,\zeta=0)]\, h_6(\xi)$$
$$+ [\frac{\partial y(\xi=0,\eta,\zeta=0)}{\partial \xi} - \frac{\partial y'_{12}(\xi=0,\eta,\zeta=0)}{\partial \xi}]\, h_7(\xi)$$
$$+ [\frac{\partial y(\xi=1,\eta,\zeta=0)}{\partial \xi} - \frac{\partial y'_{12}(\xi=1,\eta,\zeta=0)}{\partial \xi}]\, h_8(\xi) \qquad (3.20h)$$

$$\Delta z(\xi,\eta,\zeta=0) = [z(\xi=0,\eta,\zeta=0) - z'_{12}(\xi=0,\eta,\zeta=0)]\, h_5(\xi)$$
$$+ [z(\xi=1,\eta,\zeta=0) - z'_{12}(\xi=1,\eta,\zeta=0)]\, h_6(\xi)$$
$$+ [\frac{\partial z(\xi=0,\eta,\zeta=0)}{\partial \xi} - \frac{\partial z'_{12}(\xi=0,\eta,\zeta=0)}{\partial \xi}]\, h_7(\xi)$$
$$+ [\frac{\partial z(\xi=1,\eta,\zeta=0)}{\partial \xi} - \frac{\partial z'_{12}(\xi=1,\eta,\zeta=0)}{\partial \xi}]\, h_8(\xi) \qquad (3.20i)$$

$$\frac{\partial x'_{12}(\xi=0,\eta,\zeta=0)}{\partial \xi} = h_1(\eta)\, \frac{\partial x(\xi=0,\eta=0,\zeta=0)}{\partial \xi} + h_2(\eta)\, \frac{\partial x(\xi=0,\eta=1,\zeta=0)}{\partial \xi}$$
$$+ h_3(\eta) \frac{\partial^2 x(\xi=0,\eta=0,\zeta=0)}{\partial \xi \partial \eta} + h_4(\eta)\, \frac{\partial^2 x(\xi=0,\eta=1,\zeta=0)}{\partial \xi \partial \eta} \qquad (3.20j)$$

$$\frac{\partial x'_{12}(\xi=1,\eta,\zeta=0)}{\partial \xi} = h_1(\eta)\, \frac{\partial x(\xi=1,\eta=0,\zeta=0)}{\partial \xi} + h_2(\eta)\, \frac{\partial x(\xi=1,\eta=1,\zeta=0)}{\partial \xi}$$
$$+ h_3(\eta) \frac{\partial^2 x(\xi=1,\eta=0,\zeta=0)}{\partial \xi \partial \eta} + h_4(\eta)\, \frac{\partial^2 x(\xi=1,\eta=1,\zeta=0)}{\partial \xi \partial \eta} \qquad (3.20k)$$

$$\frac{\partial y'_{12}(\xi=0,\eta,\zeta=0)}{\partial \xi} = h_1(\eta)\,\frac{\partial y(\xi=0,\eta=0,\zeta=0)}{\partial \xi} + h_2(\eta)\,\frac{\partial y(\xi=0,\eta=1,\zeta=0)}{\partial \xi}$$

$$+\; h_3(\eta)\frac{\partial^2 y(\xi=0,\eta=0,\zeta=0)}{\partial \xi \partial \eta} + h_4(\eta)\,\frac{\partial^2 y(\xi=0,\eta=1,\zeta=0)}{\partial \xi \partial \eta} \qquad (3.20l)$$

$$\frac{\partial y'_{12}(\xi=1,\eta,\zeta=0)}{\partial \xi} = h_1(\eta)\,\frac{\partial y(\xi=1,\eta=0,\zeta=0)}{\partial \xi} + h_2(\eta)\,\frac{\partial y(\xi=1,\eta=1,\zeta=0)}{\partial \xi}$$

$$+\; h_3(\eta)\frac{\partial^2 y(\xi=1,\eta=0,\zeta=0)}{\partial \xi \partial \eta} + h_4(\eta)\,\frac{\partial^2 y(\xi=1,\eta=1,\zeta=0)}{\partial \xi \partial \eta} \qquad (3.20m)$$

$$\frac{\partial z'_{12}(\xi=0,\eta,\zeta=0)}{\partial \xi} = h_1(\eta)\,\frac{\partial z(\xi=0,\eta=0,\zeta=0)}{\partial \xi} + h_2(\eta)\,\frac{\partial z(\xi=0,\eta=1,\zeta=0)}{\partial \xi}$$

$$+\; h_3(\eta)\frac{\partial^2 z(\xi=0,\eta=0,\zeta=0)}{\partial \xi \partial \eta} + h_4(\eta)\,\frac{\partial^2 z(\xi=0,\eta=1,\zeta=0)}{\partial \xi \partial \eta} \qquad (3.20n)$$

$$\frac{\partial z'_{12}(\xi=1,\eta,\zeta=0)}{\partial \xi} = h_1(\eta)\,\frac{\partial z(\xi=1,\eta=0,\zeta=0)}{\partial \xi} + h_2(\eta)\,\frac{\partial z(\xi=1,\eta=1,\zeta=0)}{\partial \xi}$$

$$+\; h_3(\eta)\frac{\partial^2 z(\xi=1,\eta=0,\zeta=0)}{\partial \xi \partial \eta} + h_4(\eta)\,\frac{\partial^2 z(\xi=1,\eta=1,\zeta=0)}{\partial \xi \partial \eta} \qquad (3.20o)$$

In the above equations, h_1, h_2, h_3, and h_4 are given by equation (2.11) and h_5, h_6, h_7, and h_8 are given by equation (2.24). The method for evaluating the partial derivative terms appearing on the right hand sides of equation (3.20) is described below.

The first-order partial derivative terms in equation (3.20) determine the shape of the surface shown in figure 3-8 bounded by the four twisted curves given by equations (3.14) and (3.19). These derivative terms also determine whether grid lines on that surface will intersect the four twisted curves orthogonally or not. Since the only information we have about the surface shown in figure 3-8 is the four twisted curves, these curves are used to derive expressions for the first-order derivative terms in equation (3.20).

We shall illustrate how first-order derivative terms in equation (3.20) are calculated by deriving expressions for $\frac{\partial x(\xi,\eta=0,\zeta=0)}{\partial \eta}$, $\frac{\partial y(\xi,\eta=0,\zeta=0)}{\partial \eta}$, and $\frac{\partial z(\xi,\eta=0,\zeta=0)}{\partial \eta}$. These three derivative terms are evaluated along curve 1 in which ξ varies between 0 and 1, η=0, and ζ=0

(figs. 3-8 and 3-9). The method for deriving first-order derivative terms along the other three twisted curves (i.e., curves 2, 3, and 4 in fig. 3-8) will be similar to the procedure described below.

Expressions for the first-order partial derivative terms along curve 1 are derived in five steps and they are

1. Determine a vector which is perpendicular to the plane formed by the vectors tangent to curves 1 and 3 at one end of curve 1 (figs. 3-8 and 3-11).
2. Determine a vector which is perpendicular to the plane formed by the vectors tangent to curves 1 and 4 at the other end of curve 1 (figs. 3-8 and 3-11).
3. Linearly interpolate between the two vectors determined in Steps 1 and 2 and denote this vector function as $N_{\xi 1}$.
4. Determine a vector function that is parallel to the cross product of $N_{\xi 1}$ (Step 3) and a vector function tangent to curve 1. The vector function thus determined is assumed to be tangent to the surface that we wish to approximate along curve 1.
5. Determine first-order partial derivative terms along curve 1 by forcing the vector function formed by the first-order derivatives to be parallel to the vector function determined in Step 4.

The details of these five steps are described below.

Step 1 - Determine Normal Vector at One End of Curve 1. The vectors tangent to curves 1 and 3 at ξ=0 and η=0 are given by

$$T_{\xi 13} = \frac{\partial X_1(\xi=0)}{\partial \xi} \mathbf{I} + \frac{\partial Y_1(\xi=0)}{\partial \xi} \mathbf{J} + \frac{\partial Z_1(\xi=0)}{\partial \xi} \mathbf{K} \tag{3.21a}$$

$$T_{\eta 31} = \frac{\partial X_3(\eta=0)}{\partial \eta} \mathbf{I} + \frac{\partial Y_3(\eta=0)}{\partial \eta} \mathbf{J} + \frac{\partial Z_3(\eta=0)}{\partial \eta} \mathbf{K} \tag{3.21b}$$

where I, J, and K are unit vectors pointing in the x-, y-, and z-directions, respectively.

The vector perpendicular to the plane formed by the above two vectors is given by

$$
\begin{aligned}
N_{13} &= T_{\xi 13} \text{ X } T_{\eta 31} \\
&= \left\{ \frac{\partial Y_1(\xi=0)}{\partial \xi} \frac{\partial Z_3(\eta=0)}{\partial \eta} - \frac{\partial Z_1(\xi=0)}{\partial \xi} \frac{\partial Y_3(\xi=0)}{\partial \eta} \right\} \mathbf{I} \\
&- \left\{ \frac{\partial X_1(\xi=0)}{\partial \xi} \frac{\partial Z_3(\eta=0)}{\partial \eta} - \frac{\partial Z_1(\xi=0)}{\partial \xi} \frac{\partial X_3(\eta=0)}{\partial \eta} \right\} \mathbf{J} \\
&+ \left\{ \frac{\partial X_1(\xi=0)}{\partial \xi} \frac{\partial Y_3(\eta=0)}{\partial \eta} - \frac{\partial Y_1(\xi=0)}{\partial \xi} \frac{\partial X_3(\xi=0)}{\partial \eta} \right\} \mathbf{K}
\end{aligned} \tag{3.22}
$$

Step 2 - Determine Normal Vector at Other End of Curve 1. The vectors tangent to curves 1 and 4 at $\xi=1$ and $\eta=0$ are given by

$$T_{\xi 14} = \frac{\partial X_1(\xi=1)}{\partial \xi} \mathbf{I} + \frac{\partial Y_1(\xi=1)}{\partial \xi} \mathbf{J} + \frac{\partial Z_1(\xi=1)}{\partial \xi} \mathbf{K} \tag{3.23a}$$

$$T_{\eta 41} = \frac{\partial X_4(\eta=0)}{\partial \eta} \mathbf{I} + \frac{\partial Y_4(\eta=0)}{\partial \eta} \mathbf{J} + \frac{\partial Z_4(\eta=0)}{\partial \eta} \mathbf{K} \tag{3.23b}$$

and the vector perpendicular to the plane formed by the above two vectors is given by

$$
\begin{aligned}
N_{14} &= T_{\xi 14} \text{ X } T_{\eta 41} \\
&= \left\{ \frac{\partial Y_1(\xi=1)}{\partial \xi} \frac{\partial Z_4(\eta=0)}{\partial \eta} - \frac{\partial Z_1(\xi=1)}{\partial \xi} \frac{\partial Y_4(\xi=0)}{\partial \eta} \right\} \mathbf{I} \\
&- \left\{ \frac{\partial X_1(\xi=1)}{\partial \xi} \frac{\partial Z_4(\eta=0)}{\partial \eta} - \frac{\partial Z_1(\xi=1)}{\partial \xi} \frac{\partial X_4(\eta=0)}{\partial \eta} \right\} \mathbf{J} \\
&+ \left\{ \frac{\partial X_1(\xi=1)}{\partial \xi} \frac{\partial Y_4(\eta=0)}{\partial \eta} - \frac{\partial Y_1(\xi=1)}{\partial \xi} \frac{\partial X_4(\xi=0)}{\partial \eta} \right\} \mathbf{K}
\end{aligned} \tag{3.24}
$$

Step 3 - Linearly Interpolate between Two Normal Vectors. In this step, we linearly interpolate between the two normal vectors obtained in Steps 1 and 2 to produce the following vector function:

$$N_{\xi 1} = (1 - \xi)\, N_{13} + \xi\, N_{14} \tag{3.25}$$

where N_{13} and N_{14} are given by equations (3.22) and (3.24).

Step 4 - Determine a Vector Function Tangent to the Surface. Since there are an infinite number of surfaces that can pass through any given curve, a strategy must be developed to determine which surface is to approximate the surface shown in figure 3-8. Here, the surface selected is the one that will pass through curve 1 as well as curves 3 and 4. Thus, the vector function tangent to the surface at curve 1 can be approximated by

$$E_{\eta 1} = T_{\xi 1}\, \mathbf{X}\, N_{\xi 1} \tag{3.26}$$

where $N_{\xi 1}$ is given by equation (3.25) and $T_{\xi 1}$ is a vector function tangent to curve 1 given by

$$T_{\xi 1} = \frac{\partial X_1(\xi)}{\partial \xi}\, I + \frac{\partial Y_1(\xi)}{\partial \xi}\, \mathbf{J} + \frac{\partial Z_1(\xi)}{\partial \xi}\, \mathbf{K} \tag{3.27}$$

Step 5 - Determine Expressions for First-Order Derivatives. Now that we know along which surface partial derivatives with respect to η at curve 1 can be made, we can derive expressions for the first-order partial derivative terms $\frac{\partial x(\xi,\eta=0,\zeta=0)}{\partial \eta}$, $\frac{\partial y(\xi,\eta=0,\zeta=0)}{\partial \eta}$, and $\frac{\partial z(\xi,\eta=0,\zeta=0)}{\partial \eta}$ that appear in equation (3.20). Since we desire grid lines on the surface to be perpendicular to the boundary curve (curve 1 in this case), the vector function tangent to the η-direction, $T_{\eta 1}$, must be parallel to the vector function, $E_{\eta 1}$, derived in Step 4. This can be expressed mathematically as

$$T_{\eta 1}\, \mathbf{X}\, E_{\eta 1} = 0 \tag{3.28}$$

where

$$T_{\eta 1} = \frac{\partial x(\xi,\eta=0,\zeta=0)}{\partial \eta}\, \mathbf{I} + \frac{\partial y(\xi,\eta=0,\zeta=0)}{\partial \eta}\, \mathbf{J} + \frac{\partial z(\xi,\eta=0,\zeta=0)}{\partial \eta}\, \mathbf{K} \tag{3.29}$$

The desired expressions for the first-order partial derivative terms are obtained from equation (3.28) in a manner very similar to the derivation of equation (2.12). Again, "K factors" which appear in equation (2.12) can also be used here to ensure that grid lines do not overlap each other and to create a smoother surface.

3.3.3 Parametric Bihyperbolic Spline Interpolation. — To illustrate how parametric bihyperbolic spline interpolation approximates surfaces, consider the organized set of $IM \times JM$ nodal points shown in figure 3-6. These points lie on surface H and are given on a rectangular xy-grid. We will call this collection of nodal points N and represent each nodel point within N by

$$N(i,j) = \{x_{ij},\ y_{ij},\ z_{ij}\}\ ,\quad i = 1, 2, \ldots, IM,\qquad j = 1, 2, \ldots, JM \tag{3.30}$$

We wish to obtain a parametric description of surface H by interpolating between the nodal points of N. We will accomplish this in three steps. First, we will calculate approximate arc lengths along lines of constant i and j. Next, we will determine values of the derivatives at the nodal points which lie along the edges of surface H. Finally, we will use bihyperbolic spline interpolation to interpolate between the nodal points using our approximate arc lengths as the independent variables. This approach is similar to that suggested by Smith (ref. 36).

We begin the first step by denoting approximate arc length along a line of constant i as s and approximate arc length along a line of constant j as t. We calculate s and t for each nodal point using the following expressions:

$$s_{11} = t_{11} = 0 \tag{3.31}$$

$$s_{ij} = s_{i\text{-}1\ j} + \sqrt{(x_{ij}-x_{i\text{-}1\ j})^2 + (y_{ij}-y_{i\text{-}1\ j})^2 + (z_{ij}-z_{i\text{-}1\ j})^2} \tag{3.32a}$$

$$t_{ij} = t_{i\ j\text{-}1} + \sqrt{(x_{ij}-x_{i\ j\text{-}1})^2 + (y_{ij}-y_{i\ j\text{-}1})^2 + (z_{ij}-z_{i\ j\text{-}1})^2} \tag{3.32b}$$

where

$$i = 2, 3, \ldots, IM, \qquad j = 2, 3, \ldots, JM$$

Here, s_{ij} and t_{ij} are the s and t values at nodal point $N(i,j)$.

Having accomplished this, the next step is to calculate values for the derivatives at the nodal points which lie along the edges of surface H. This information is required by the bihyperbolic spline interpolation method. The derivatives that are required are as follows:

$$\frac{\partial x}{\partial s}, \frac{\partial y}{\partial s}, \text{ and } \frac{\partial z}{\partial s} \quad \text{at } N(i,j), \quad i = 1, IM, \quad j = 1, 2, \ldots, JM \tag{3.33a}$$

$$\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t}, \text{ and } \frac{\partial z}{\partial t} \quad \text{at } N(i,j), \quad i = 1, 2, \ldots, IM, \quad j = 1, JM \tag{3.33b}$$

$$\frac{\partial^2 x}{\partial s\, \partial t}, \frac{\partial^2 y}{\partial s\, \partial t}, \text{ and } \frac{\partial^2 z}{\partial s\, \partial t} \quad \text{at } N(1,1),\ N(IM,1),\ N(1,JM) \text{ and } N(IM,JM) \tag{3.33c}$$

We recommend using second-order-accurate finite-difference formulas to calculate these values from the coordinates of the nodal point coordinates. A comprehensive list of finite-difference approximations for derivatives appears in reference 60.

Once we obtain values for the derivatives in equation (3.33), we are now ready to use bihyperbolic spline interpolation to form an approximation to surface H. Here, we follow closely Späth's description of bihyperbolic spline interpolation found in reference 50. In bihyperbolic spline interpolation, we must determine the coefficients a_{ijkl} of the expression

$$u(s,t) = \sum_{k=1}^{4} \sum_{l=1}^{4} a_{ijkl}\, \phi_k(s,\alpha_i,s_{ij},s_{i+1\,j})\, \phi_l(t,\beta_j,t_{ij},t_{i+1\,j}) \tag{3.34}$$

where u represents either x, y, or z in the Cartesian system. The function $u(s,t)$ in equation (3.34) must take on the specified values of x, y, or z (depending on which one it represents) at

the appropriate nodal point locations while maintaining continuity up to the second-order derivative everywhere. The functions ϕ_k and ϕ_l are given by

$$\phi_1(s, \alpha_i, s_{ij}, s_{i+1\,j}) = s - s_{ij} \tag{3.35a}$$

$$\phi_2(s, \alpha_i, s_{ij}, s_{i+1\,j}) = s_{i+1\,j} - s, \tag{3.35b}$$

$$\phi_3(s, \alpha_i, s_{ij}, s_{i+1\,j}) = \psi(s - s_{ij}, \alpha_i, s_{ij}, s_{i+1\,j}) \tag{3.35c}$$

$$\phi_4(s, \alpha_i, s_{ij}, s_{i+1\,j}) = \psi(s_{i+1\,j} - s, \alpha_i, s_{ij}, s_{i+1\,j}) \tag{3.35d}$$

where

$$\psi(s, \alpha_i, s_{ij}, s_{i+1\,j}) = \frac{\Delta s_{ij} sinh(\alpha_i s) - s\ sinh(\alpha_i \Delta s_{ij})}{sinh(\alpha_i \Delta s_{ij}) - \alpha_i \Delta s_{ij}} \tag{3.35e}$$

$$\Delta s_{ij} = s_{i+1\,j} - s_{ij} \tag{3.35f}$$

Here, α_i and β_j are tension parameters in the s and t "directions," respectively. They are constrained by $0 < \alpha_i < \infty$ and $0 < \beta_j < \infty$ with tension increasing as they approach infinity. Each grid cell (fig. 3-6) can have different α_i and β_j values depending on the amount of tension desired. Here, we note that as α_i and β_j approach zero, the bihyperbolic spline surface approaches a bicubic spline surface (ref. 50).

The reader may recognize that equation (3.35e) contains expressions similar to those in equation (3.12) of Section 3.2.2. In fact, equation (3.12) can be cast in the form

$$X_i(s) = \sum_{k=1}^{4} A_{ik}\ \phi_k(s, \alpha_i, s_i, s_{i+1}) \tag{3.36}$$

where ϕ_k is given by equation (3.35), and the A_{ik}'s are the coefficients which contain the unknown $X''(s_i)$ values. Thus, the bihyperbolic spline described by equation (3.34) is merely an extension of the tension spline of Section 3.2.2.

We begin the algorithm for obtaining the coefficients, a_{ijkl}, by letting $u_{ij} = u(s_{ij}, t_{ij})$, $p_{ij} = \frac{\partial u_{ij}}{\partial s}$, $q_{ij} = \frac{\partial u_{ij}}{\partial t}$, and $r_{ij} = \frac{\partial^2 u_{ij}}{\partial s\, \partial t}$. Recall that u_{ij} is known at all nodal points, while p_{ij}, q_{ij}, and r_{ij} are only known at the nodal points which lie on the boundaries of H. Values of p_{ij},

q_{ij}, and r_{ij} at nodal points other than those on the boundaries can be determined by solving the following $2(IM) + JM + 2$ linear system of equations:

$$b_{i-1\,j}\,p_{i-1\,j} + (b_{i-1\,j}\,v_{i-1\,j} + b_{ij}\,v_{ij})\,p_{ij} + b_{ij}\,p_{i+1\,j}$$

$$= b_{i-1\,j}\,(v_{i-1\,j} + 1)\frac{u_{ij}-u_{i-1\,j}}{\Delta s_{i-1\,j}} + b_{ij}(v_{ij} + 1)\frac{u_{i+1\,j}-u_{ij}}{\Delta s_{ij}}$$

$$\text{for } j = 1, 2, \ldots, JM\,, \quad i = 2, 3, \ldots, IM-1 \tag{3.37a}$$

$$c_{i\,j-1}\,q_{i\,j-1} + (c_{i\,j-1}\,w_{i\,j-1} + c_{ij}\,w_{ij})\,q_{ij} + c_{ij}\,q_{i\,j+1}$$

$$= c_{i\,j-1}\,(w_{i\,j-1} + 1)\frac{u_{ij}-u_{i\,j-1}}{\Delta t_{i\,j-1}} + c_{ij}(w_{ij} + 1)\frac{u_{i\,j+1}-u_{ij}}{\Delta t_{ij}}$$

$$\text{for } i = 1, 2, \ldots, IM\,, \quad j = 2, 3, \ldots, JM-1 \tag{3.37b}$$

$$b_{i-1\,j}\,r_{i-1\,j} + (b_{i-1\,j}\,v_{i-1\,j} + b_{ij}\,v_{ij})\,r_{ij} + b_{ij}\,r_{i+1\,j}$$

$$= b_{i-1\,j}\,(v_{i-1\,j} + 1)\frac{q_{ij}-q_{i-1\,j}}{\Delta s_{i-1\,j}} + b_{ij}(v_{ij} + 1)\frac{q_{i+1\,j}-q_{ij}}{\Delta s_{ij}}$$

$$\text{for } j = 1, JM\,, \quad i = 2, 3, \ldots, IM-1 \tag{3.37c}$$

$$c_{i\,j-1}\,r_{i\,j-1} + (c_{i\,j-1}\,w_{i\,j-1} + c_{ij}\,w_{ij})\,r_{ij} + c_{ij}\,r_{i\,j+1}$$

$$= c_{i\,j-1}\,(w_{i\,j-1} + 1)\frac{p_{ij}-p_{i\,j-1}}{\Delta t_{i\,j-1}} + c_{ij}(w_{ij} + 1)\frac{p_{i\,j+1}-p_{ij}}{\Delta t_{ij}}$$

$$\text{for } i = 1, 2, \ldots, IM\,, \quad j = 2, 3, \ldots, JM-1 \tag{3.37d}$$

Here, v_{ij}, w_{ij}, b_{ij}, and c_{ij} are given by

$$v_{ij} = \left(\frac{d}{dx}\psi(s-s_{ij},\, \alpha_i,\, s_{ij},\, s_{i+1\,j})\right)_{s=s_{i+1\,j}} (i = 1, 2, \ldots, IM-1) \tag{3.38a}$$

$$w_{ij} = \left(\frac{d}{dx}\psi(t-t_{ij},\, \beta_i,\, t_{ij},\, t_{i+1\,j})\right)_{t=t_{i\,j+1}} (j = 1, 2, \ldots, JM-1) \tag{3.38b}$$

$$b_{ij} = \frac{{\alpha_i}^2 \Delta s_{ij} sinh(\alpha_i \Delta s_{ij})}{({v_{ij}}^2-1)(sinh(\alpha_i \Delta s_{ij})-\alpha_i \Delta s_{ij})} \qquad (i = 1, 2, \ldots, IM-1) \tag{3.38c}$$

$$c_{ij} = \frac{{\beta_j}^2 \Delta t_{ij} sinh(\beta_i \Delta t_{ij})}{({w_{ij}}^2-1)(sinh(\beta_i \Delta t_{ij})-\beta_i \Delta t_{ij})} \qquad (j = 1, 2, \ldots, JM-1) \tag{3.38d}$$

The system of equations described by equations (3.37) and (3.38) is obtained by requiring that the function u and its partial derivatives p, q, and r be continuous at all of the interior nodal points in N. This system has a tridiagonal coefficient matrix that can be shown to be diagonally dominant (ref. 50). Thus, the system may be easily solved for the unknown values of p_{ij}, q_{ij}, and r_{ij} by using the Thomas algorithm.

We now solve for the a_{ijkl}'s of equation (3.34) by first defining a 4×4 coefficient matrix within each grid cell as

$$A_{ij} = a_{ijkl} \ (k, \ l = 1, 2, 3, 4) \tag{3.39}$$

This matrix can be found from the following matrix product expression (ref. 50):

$$A_{ij} = C(\Delta s_{ij}, v_{ij}) \ K_{ij}[C(\Delta t_{ij}, w_{ij})]^T \tag{3.40a}$$

The variables C and K_{ij} are matrix expressions which are given by the following:

$$C(g,h) = \begin{bmatrix} 0 & 0 & 1/g & 0 \\ 1/g & 0 & 0 & 0 \\ -\dfrac{1}{g(1-h)} & \dfrac{-1}{1-h^2} & \dfrac{1}{g(1-h)} & \dfrac{-h}{1-h^2} \\ \dfrac{1}{g(1-h)} & \dfrac{h}{1-h^2} & -\dfrac{1}{g(1-h)} & \dfrac{1}{1-h^2} \end{bmatrix} \tag{3.40b}$$

$$K_{ij} = \begin{bmatrix} u_{ij} & q_{ij} & u_{i\,j+1} & q_{i\,j+1} \\ p_{ij} & r_{ij} & p_{i\,j+1} & r_{i\,j+1} \\ u_{i+1\,j} & q_{i+1\,j} & u_{i+1\,j+1} & q_{i+1\,j+1} \\ p_{i+1\,j} & r_{i+1\,j} & p_{i+1\,j+1} & r_{i+1\,j+1} \end{bmatrix} \tag{3.40c}$$

Having found the coefficients a_{ijkl}, we can now use equation (3.35) to calculate $u(s,t)$ at any s and t.

Returning to the problem of obtaining a description of surface H, we perform three bihyperbolic interpolations (one for $u=x(s,t)$, one for $u=y(s,t)$, and one for $u=z(s,t)$) using the procedure above. In this way, we obtain the parametric representation of surface H that is necessary for the grid generation technique presented in Section 2.0.

4.0 GENERATION OF SINGLE AND COMPOSITE GRIDS

In the previous two sections, the details of the algebraic grid generation methods used in GRID2D/3D are described. In this section, we demonstrate the usefulness of GRID2D/3D by using it to generate a number of single and composite grids within complex-shaped 2- and 3-D spatial domains. Recall from Section 1.0, a single grid is a grid system based on one boundary-fitted coordinate system, and a composite grid is a grid system made up of two or more single grids patched together.

4.1 Single Grids

All of the details involved in using the Two-, Four-, and Six-Boundary Methods to generate single grids were presented in Sections 2.0 and 3.0. In fact, while illustrating these grid generation methods, several single grids were generated (figs. 2-3, 2-4, and 2-8). In this subsection, several additional single grids generated by GRID2D/3D are presented to illustrate the capabilities of GRID2D/3D. Also presented in this section is a brief discussion on how to smooth discontinuities in grid systems that arise from boundary discontinuities.

Figure 4-1 shows a 2-D single grid generated for an irregular, 2-D, coastline topology using the Two-Boundary Method. A stretching function was used to cluster grid points near the coastline where complex flow conditions are expected. The coastline curve was formed by using parametric tension spline interpolation.

Figure 4-2 shows a 2-D single grid around a sharp bend generated by using the Two-Boundary Method. This figure illustrates how boundary discontinuities can propagate into the interior of the grid. The slope discontinuity of the η grid lines along $i = I_1$ can be eliminated by applying the following equations:

$$y_{i,j}^{n+1} = y_{i,j}^{n} + .25\ (y_{i+1,j}^{n} - 2\ y_{i,j}^{n} + y_{i-1,j}^{n}) \tag{4.1a}$$

$$x_{i,j}^{n+1} = x_{i,j}^{n} + .25\ (x_{i+1,j}^{n} - 2\ x_{i,j}^{n} + x_{i-1,j}^{n}) \tag{4.1b}$$

where i = I_1−m,...,I_1,...,I_1+m (m can be zero or some positive integer constant) and j = 2,3,...,JL−1. In order to smooth the grid shown in figure 4-2, equation (4.1) needs to be applied a number of times; that is, n = 0, 1, 2, 3, ... with n = 0 being the grid shown in figure 4-2, n = 1 being the first correction, n = 2 being the second correction, and so on. By using equation (4.1) repeatedly, the grid shown in figure 4-2 was smoothed as shown in figure 4-3.

Figure 4-4 shows a 2-D single grid around an airfoil. This grid was generated by using the Two-Boundary Method in which stretching functions were used to cluster grid points near the airfoil surface in anticipation of the complex boundary layer flow there.

Figure 4-5 shows a 2-D single grid for an irregularly shaped, four-sided region. That figure clearly shows how orthogonality of the grid lines is enforced at each of the four boundaries of the spatial domain. This grid was generated by using the Four-Boundary Method and stretching functions were not used. The four boundary curves of the spatial domain shown in figure 4-5 were represented in parametric form by tension spline interpolation.

A 3-D single grid between the twisted blades of a radial turbine is shown in figure 4-6. The Two-Boundary Method was used to generate this grid and a stretching function was used to cluster grid points near the blade surfaces. The blade surfaces were generated by using tension splines in conjunction with bilinearly blended transfinite interpolation.

As an example of the efficiency of GRID2D/3D, the typical real time taken to generate a 21 × 21 × 21 3-D single grid is 45 seconds on an IBM AT compatible personal computer running at 12 MHz.

4.2 Composite Grids

In Section 1.1, it was noted that GRID2D/3D can be used to generate composite grids as well as single grids. Also, it was noted that composite grids which can be generated by GRID2D/3D include those that are completely discontinuous, partially discontinuous, and partially continuous (fig. 1-2). For completely or partially discontinuous composite grids such as those shown in figures. 1-2(a) and 1-2(b), each single grid within the composite grid can be different from any other in structure and in the number of grid points. Thus, each single grid within such composite grids can be generated independently of the other single grids, and each single grid can be generated in the manner described in Sections 2.0 and 4.1. How the different single grids of such composite grids are patched together once they are generated is described in Sections 1.1.1 and 1.1.2. Since discontinuous composite grids can be generated rather easily and patching is trivial, no further discussions concerning them will be given.

For partially continuous composite (PCC) grids, each single grid within it must satisfy a number of compatibility conditions so that when the different single grids are patched together, the resultant composite grid will have some degree of continuity (Section 1.1.3). PCC grids such as the one shown in figure 1-2(d) are often difficult to generate. However, when it is possible to generate them, they are the easiest to use with FD and FV methods to obtain solutions to partial differential equations. Below, we discuss how PCC grids can be generated.

4.3 Partially Continuous Composite Grids

PCC grids are generated by GRID2D/3D in two major steps. The first step involves partitioning the spatial domain into zones. The second step involves constructing grid systems

that will be continuous from one zone to another. These two steps are described in detail below.

4.3.1 Partitioning. – The first step in constructing a PCC grid is to partition the spatial domain of interest into a finite number of contiguous zones. The partitioning process involves answering three interrelated questions:

1. How should the spatial domain be partitioned into zones?
2. Based on that partition, what grid structure is to be used within each zone?
3. Based on that partition and grid structure selections, how should the different zones be mapped to the transformed domain so that the resultant composite grid will have some degree of continuity?

The answers to these questions depend on the geometry of the spatial domain and the physics of the problem for which the grid is being generated. For any given problem, several different choices are usually possible. However, for simplicity, the choice containing the least number of zones is often the most desirable.

An example illustrating one strategy at partitioning is shown in figure 1-3(a). The boundaries of that spatial domain contain a backward facing step, a flat wall, and a wedge-shaped obstacle. The fluid is flowing from left to right. In figure 1-3, the spatial domain is partitioned by using free-streamline theory. More specifically, a new zone is set up where separation is expected (e.g., at the backward step and at the rear of the wedge-shaped obstacle) or where a flow will separate into two streams (e.g., at the nose of the wedge-shaped obstacle). The structure of the grid within each zone in figure 1-3(a) was chosen to be H-type. This structure aligns the main flow direction with the grid lines which is important because it reduces dissipation error and permits the use of the thin-layer Navier-Stokes equations. Also, this structure allows grid lines to be clustered near solid wall boundaries readily as well as

allows continuity of grid lines and their derivatives from one zone to another. Figure 1-3(b) shows how the different zones in figure 1-3(a) are mapped to the transformed domain.

As a second example, consider the 2-D inlet of an aircraft engine shown in figure 4-7. This geometry would be awkward to handle with a single grid due to the separation of the internal flow from the external flow by the cowl. Figure 4-8 shows a partitioning of this spatial domain into two zones and the transformed domain to which it is mapped. Other configurations involving more than two zones are certainly possible, but more complicated configurations are unnecessary unless the physics of the flow dictates their use. If we knew something about the flowfield within this spatial domain, we might wish to introduce new zones in order to better resolve the physical aspects of the flow (e.g., to make capturing of shock waves more convenient). In some cases, we might even want to solve different governing equations in different zones in order to obtain solutions more rapidly.

As a third example, consider the spatial domain shown in figure 4-9. This is a 2-D region near a wind tunnel wall with slots cut into it. Two possible partitions for this spatial domain are shown in figure 4-10. The partition shown in figure 4-10(a) contains six zones. A better partition with only three zones is shown in figure 4-10(b).

4.3.2 Grid Generation with Patching. – Once we have partitioned the spatial domain of interest into zones, we are ready to generate a grid for each zone. For partially continuous composite (PCC) grids, the grid generated for each zone must be such that when they are patched together, the resultant composite grid has some degree of continuity. It turns out that this requirement necessitates some minor modifications to the Two-, Four-, and Six-Boundary Methods presented in Section 2.0. In this section, these modifications are described in the framework of the Two-Boundary Method by applying it to generate a PCC grid in the 2-D inlet of the aircraft engine shown in figure 4-7.

Step 1 - Define the Coordinate Transformation. We seek a coordinate transformation of the form given by equation (2.19).

Step 2 - Select a Time-Stretching Function. We set τ equal to t according to equation (2.2).

Step 3 - Select Two Boundaries of Each Zone. We choose curves 1 and 2 for zone 1 and curves 5 and 6 for zone 2 (fig. 4-8). For each zone, we map these two curves to coordinate lines $\eta=\eta_{low}$ and $\eta=\eta_{high}$, respectively. Here, η_{low}, η_{high}, ξ_{low}, and ξ_{high} are defined as the coordinate lines in the transformed domain which correspond to the boundaries of a zone. Thus, for zone 1 we have

$$X_1 = x(\xi,\eta=\eta_{low}) = X_1(\xi) \tag{4.2a}$$

$$Y_1 = y(\xi,\eta=\eta_{low}) = Y_1(\xi) \tag{4.2b}$$

$$X_2 = x(\xi,\eta=\eta_{high}) = X_2(\xi) \tag{4.2c}$$

$$Y_2 = y(\xi,\eta=\eta_{high}) = Y_2(\xi) \tag{4.2d}$$

where $\eta_{low}=0$ and $\eta_{high}=0.5$. X_1 and Y_1 are the x- and y-coordinates of curve 1, and X_2 and Y_2 are the x- and y-coordinates of curve 2.

For zone 2, we have

$$X_5 = x(\xi,\eta=\eta_{low}) = X_5(\xi) \tag{4.3a}$$

$$Y_5 = y(\xi,\eta=\eta_{low}) = Y_5(\xi) \tag{4.3b}$$

$$X_6 = x(\xi,\eta=\eta_{high}) = X_6(\xi) \tag{4.3c}$$

$$Y_6 = y(\xi,\eta=\eta_{high}) = Y_6(\xi) \tag{4.3d}$$

where $\eta_{low}=0.5$ and $\eta_{high}=1$. X_5 and Y_5 are the x- and y- coordinates of curve 5, and X_6 and Y_6 are the x- and y-coordinates of curve 6.

The other two curves in each zone (curves 3 and 4 in zone 1, and curves 7 and 8 in zone 2) are mapped to coordinate lines $\xi=\xi_{low}$ and $\xi=\xi_{high}$ in the transformed domain. For our example, $\xi_{low}= 0$ and $\xi_{high}= 1$ for both zones 1 and 2.

If we were using the Four- or the Six-Boundary Method, then we would have selected four or six boundaries in each zone as described in Sections 2.2 and 2.3.

Step 4 - Describe the Two Boundaries of Each Zone in Parametric Form. We now represent the four curves selected in Step 3 (two for each zone) in parametric form. Here, information about the four curves is given in the form of four sets of discrete points which lie along the curves. Thus, we use the 2-D parametric tension spline interpolation method described in Section 3.2.2 to generate the parametric equations in the form dictated by equations (4.2) and (4.3). By using this method, we generate the 2-D curve approximations shown in figure 4-11.

Step 5 - Define Curves That Connect the Two Boundaries in Each Zone. As before, we use transfinite interpolation to derive curves which connect the two boundaries of each zone described in Step 3. For Lagrange interpolation, the equations for the connecting curves between two boundaries in any one zone are

$$x(\xi,\eta) = x(\xi,\eta=\eta_{low})\ l_1(\eta) + x(\xi,\eta=\eta_{high})\ l_2(\eta) \tag{4.4a}$$

$$y(\xi,\eta) = y(\xi,\eta=\eta_{low})\ l_1(\eta) + y(\xi,\eta=\eta_{high})\ l_2(\eta) \tag{4.4b}$$

where the blending functions, l_1 and l_2, are given by

$$l_1(\eta) = \frac{\eta-\eta_{high}}{\eta_{low}-\eta_{high}} \tag{4.4c}$$

$$l_2(\eta) = \frac{\eta-\eta_{low}}{\eta_{high}-\eta_{low}} \tag{4.4d}$$

As stated in Section 2.1, the curves described by equation (4.4) are straight lines which do not, in general, intersect the boundary curves perpendicularly. This could lead to slope discontinuities in grid lines which cross zonal interfaces (fig. 4-12). For this reason, we recommend the use of transfinite interpolation based on Hermite interpolation when patching. Hermite interpolation allows the connecting curves to intersect the boundary curves perpendicularly. Thus, grid lines in adjacent zones which meet at the zonal interface will possess identical slopes at the interface. This eliminates slope discontinuities of the grid lines at zonal interfaces.

For Hermite interpolation, the functional form for the connecting curves between two selected boundaries of a zone is as follows:

$$x(\xi,\eta) = x(\xi,\eta=\eta_{low})\, h_1(\eta) + x(\xi,\eta=\eta_{high})\, h_2(\eta) + \frac{\partial x(\xi,\eta=\eta_{low})}{\partial \eta}\, h_3(\eta) + \frac{\partial x(\xi,\eta=\eta_{low})}{\partial \eta}\, h_4(\eta) \tag{4.5a}$$

$$y(\xi,\eta) = y(\xi,\eta=\eta_{low})\, h_1(\eta) + y(\xi,\eta=\eta_{high})\, h_2(\eta) + \frac{\partial y(\xi,\eta=\eta_{low})}{\partial \eta}\, h_3(\eta) + \frac{\partial y(\xi,\eta=\eta_{low})}{\partial \eta}\, h_4(\eta) \tag{4.5b}$$

In the above equations, h_1, h_2, h_3, and h_4 are constrained by

$$h_1(\eta = \eta_{low}) = 1 \qquad h_1(\eta = \eta_{high}) = 0$$

$$\frac{\partial h_1(\eta = \eta_{low})}{\partial \eta} = 0 \qquad \frac{\partial h_1(\eta = \eta_{high})}{\partial \eta} = 0$$

$$h_2(\eta = \eta_{low}) = 0 \qquad h_2(\eta = \eta_{high}) = 1$$

$$\frac{\partial h_2(\eta = \eta_{low})}{\partial \eta} = 0 \qquad \frac{\partial h_2(\eta = \eta_{high})}{\partial \eta} = 0$$

$$h_3(\eta = \eta_{low}) = 0 \qquad h_3(\eta = \eta_{high}) = 0$$

$$\frac{\partial h_3(\eta = \eta_{low})}{\partial \eta} = 1 \qquad \frac{\partial h_3(\eta = \eta_{high})}{\partial \eta} = 0$$

$$h_4(\eta = \eta_{low}) = 0, \qquad h_4(\eta = \eta_{high}) = 0$$

$$\frac{\partial h_4(\eta = \eta_{low})}{\partial \eta} = 0, \qquad \frac{\partial h_4(\eta = \eta_{high})}{\partial \eta} = 1$$

With these constraints, h_1, h_2, h_3, and h_4 become

$$h_1 = a_1\eta^3 + b_1\eta^2 + c_1\eta + d_1 \tag{4.6a}$$
$$h_2 = a_2\eta^3 + b_2\eta^2 + c_2\eta + d_2 \tag{4.6b}$$
$$h_3 = a_3\eta^3 + b_3\eta^2 + c_3\eta + d_3 \tag{4.6c}$$
$$h_4 = a_4\eta^3 + b_4\eta^2 + c_4\eta + d_4 \tag{4.6d}$$

where the following expressions hold:

$$a_1 = \frac{2}{3(\eta_{low}^2-\eta_{high}^2)(\eta_{low}-\eta_{high})+2(3\eta_{low}^2\eta_{high}-2\eta_{low}^3-\eta_{high}^3)} \tag{4.6e}$$

$$b_1 = \frac{-3a_1(\eta_{low}^2-\eta_{high}^2)}{2(\eta_{low}-\eta_{high})} \tag{4.6f}$$

$$c_1 = -3\eta_{low}^2 a_1 - 2\eta_{low} b_1 \tag{4.6g}$$
$$d_1 = -\eta_{high}^3 a_1 - \eta_{high}^2 b_1 - \eta_{high} c_1 \tag{4.6h}$$
$$a_2 = -a_1 \tag{4.6i}$$
$$b_2 = -b_1, \tag{4.6j}$$
$$c_2 = -c_1, \tag{4.6k}$$
$$d_2 = 1 - d_1 \tag{4.6l}$$

$$a_3 = \frac{-(\eta_{low}-\eta_{high})}{3(\eta_{low}^2-\eta_{high}^2)(\eta_{low}-\eta_{high})+2(3\eta_{low}^2\eta_{high}-2\eta_{low}^3-\eta_{high}^3)} \tag{4.6m}$$

$$b_3 = \frac{1-3a_3(\eta_{low}^2-\eta_{high}^2)}{2(\eta_{low}-\eta_{high})} \tag{4.6n}$$

$$c_3 = 1 - 3\eta_{low}^2 a_3 - 2\eta_{low} b_3 \tag{4.6o}$$
$$d_3 = -\eta_{high}^3 a_3 - \eta_{high}^2 b_3 - \eta_{high} c_3 \tag{4.6p}$$

$$a_4 = a_3 \tag{4.6q}$$

$$b_4 = \frac{1-3a_4(\eta_{low}^2-\eta_{high}^2)}{2(\eta_{low}-\eta_{high})} \tag{4.6r}$$

$$c_4 = 1 - 3\eta_{low}^2 a_4 - 2\eta_{low} b_4 \tag{4.6s}$$

$$d_4 = -\eta_{high}^3 a_4 - \eta_{high}^2 b_4 - \eta_{high} c_4 \tag{4.6t}$$

Similar to the examples in Section 2.0, the partial derivative terms in equation (4.5) are chosen so that the connecting curves will intersect the two boundary curves perpendicularly. Thus, for zone 1 we have

$$\frac{\partial x(\xi,\eta=\eta_{low})}{\partial \eta} = -K_{11}(\xi)\frac{\partial Y_1(\xi)}{\partial \xi} \tag{4.7a}$$

$$\frac{\partial y(\xi,\eta=\eta_{low})}{\partial \eta} = K_{11}(\xi)\frac{\partial X_1(\xi)}{\partial \xi} \tag{4.7b}$$

$$\frac{\partial x(\xi,\eta=\eta_{high})}{\partial \eta} = -K_{12}(\xi)\frac{\partial Y_2(\xi)}{\partial \xi} \tag{4.7c}$$

$$\frac{\partial y(\xi,\eta=\eta_{high})}{\partial \eta} = K_{12}(\xi)\frac{\partial X_2(\xi)}{\partial \xi} \tag{4.7d}$$

where $\eta_{low} = 0$ and $\eta_{high} = 0.5$. Likewise, for zone 2 we have

$$\frac{\partial x(\xi,\eta=\eta_{low})}{\partial \eta} = -K_{21}(\xi)\frac{\partial Y_5(\xi)}{\partial \xi} \tag{4.8a}$$

$$\frac{\partial y(\xi,\eta=\eta_{low})}{\partial \eta} = K_{21}(\xi)\frac{\partial X_5(\xi)}{\partial \xi} \tag{4.8b}$$

$$\frac{\partial x(\xi,\eta=\eta_{high})}{\partial \eta} = -K_{22}(\xi)\frac{\partial Y_6(\xi)}{\partial \xi} \tag{4.8c}$$

$$\frac{\partial y(\xi,\eta=\eta_{high})}{\partial \eta} = K_{22}(\xi)\frac{\partial X_6(\xi)}{\partial \xi} \tag{4.8d}$$

where $\eta_{low} = 0.5$ and $\eta_{high} = 1.0$.

An alternative procedure to the one just described for calculating $x(\xi,\eta)$ and $y(\xi,\eta)$ is to do two mappings for each zone. The first mapping is from (x,y) to (ξ',η') with boundaries of both ξ' and η' between 0 and 1. The second mapping is from (ξ',η') to (ξ,η) with the

boundaries of ξ between $\xi=\xi_{low}$ and $\xi=\xi_{high}$ and the boundaries of η between $\eta=\eta_{low}$ and $\eta=\eta_{high}$. The first mapping can be accomplished by using the equations in Section 2.0 with ξ and η replaced by ξ' and η'. The second mapping is given by $\xi = \xi_{low} + \xi' \ (\xi_{high} - \xi_{low})$ and $\eta = \eta_{low} + \eta' \ (\eta_{high} - \eta_{low})$.

Step 6 - Discretize the Domain. We discretize the domain in the ξ-η-τ coordinate system by replacing the temporal domain with equally incremented time levels and by replacing each zone of the spatial domain with equally spaced grid points (fig. 4-13(b)). The time levels are described by equation (2.14). For our two-zone example, the grid points in zone 1 are located at (ξ_i,η_j), where

$$\xi_i = (i-1)\ \Delta\xi\ , \qquad i=1, 2, \ldots, IL, \tag{4.9a}$$

$$\eta_j = (j-1)\ \Delta\eta\ , \qquad j=1, 2, \ldots, JL1, \tag{4.9b}$$

and

$$\Delta\xi = \frac{1}{IL-1} \qquad\qquad \Delta\eta = \frac{1}{JL-1} \tag{4.9c}$$

The grid points in zone 2 are located at (ξ_i,η_j), where

$$\xi_i = (i-1)\ \Delta\xi\ , \quad i= 1, 2, \ldots, IL \tag{4.9d}$$

$$\eta_j = (j-1)\ \Delta\eta\ , \quad j= JL1, JL1+1, \ldots, JL \tag{4.9e}$$

By substituting equation (4.9) into equation (4.5), we obtain the locations of the grid points in the x-y-t coordinate system (fig. 4-13(a)).

We note that grid points $IL1$ through IL along grid line $JL1$ in the transformed domain represent two grid points in the spatial domain. When using this grid to obtain numerical solutions, special care must be exercised to ensure that the correct grid point is used at the appropriate situation.

Step 7 - Control the Distribution of Grid Points in Each Zone. For high-speed flows through the inlet, we expect large velocity gradients next to all solid surfaces. Thus, grid points need to be clustered near solid surfaces. With this in mind, we use stretching functions to cluster grid points near curves 1 and 2 for zone 1 and near curve 5 for zone 2. The new distribution of grid points is shown in figure 4-14.

Here, we make a few comments about the continuity of grid lines and grid spacings across zonal interfaces. In the above example, grid lines remained continuous along the portion of the zonal interface which does not represent a "physical" boundary. This is a desirable situation since it simplifies the numerical algorithm which will be used to investigate the flow in the spatial domain. Proper allignment of grid lines at the zonal interface in this case is made possible by using an appropriate stretching function in each zone. Thus, one should exercise care when constructing a grid to ensure that grid lines remain continuous across the sections of the zonal interfaces which touch each other in the spatial domain. For our example, we also note that the grid spacing in the η-direction must not change too abruptly across the zonal interface. Grid spacing in the η-direction is defined as the distance between adjacent grid points along a grid line of constant ξ. As with grid alignment, proper grid spacing at the zonal interface depends upon using an appropriate stretching function in each zone.

Step 8 - Calculate Metric Coefficients. For the spatial domain shown in figure 4-7, the five metric coefficients which need to be evaluated are τ_t, ξ_x, η_x, ξ_y, and η_y. These metric coefficients can be determined by using equation (2.28). We note that one-sided differencing should be used when calculating partial derivative terms at grid points which lie along the portions of the zonal interface which do not touch in the spatial domain. Central differencing can be used at the other grid point locations along the zonal interface.

5.0 SUMMARY

A computer program – referred to as GRID2D/3D – has been developed to generate single and composite grid systems in complex-shaped 2- and 3-D spatial domains. This technical memorandum describes the details of the algebraic grid generation methods used in GRID2D/3D, namely, the Two-, Four-, and Six-Boundary Methods. This technical memorandum also describes the methods used to derive parametric representations of curves and surfaces needed by the grid generation techniques. Several grid systems generated by GRID2D/3D for various 2- and 3-D spatial domains were presented to illustrate that GRID2D/3D is efficient and capable of handling complex geometries.

ACKNOWLEDGEMENT

This research was supported by NASA grants NAG3-929 and NAG3-997. The authors are grateful to NASA Lewis Research Center for this support.

REFERENCES

1. Smith, R.E., ed.: Numerical Grid Generation Techniques. NASA CP-2166, 1980.
2. Thompson, J.F., ed.: Numerical Grid Generation. Elsevier Science Publishing Co., 1982.
3. Thompson, J.F.; Warsi, Z.U.A.; and Mastin, C.W.: Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations - a Review. J. Comput. Phys., vol. 47, July 1982, pp. 1-108.
4. Babuska, I.; Chandra, J.; and Flaherty, J.E., eds.: Adaptive Computational Methods for Partial Differential Equations. SIAM, Philadelphia, PA, 1983.
5. Ghia, K.N.; and Ghia, U., eds: Advances in Grid Generation. FED-Vol. 5, ASME, New York, 1983.
6. Thompson, J.F.: Grid Generation Techniques in Computational Fluid Dynamics. AIAA J., vol. 22, Nov. 1984, pp. 1505-1523.
7. Eiseman, P.: Grid Generation for Fluid Mechanics Computations. Annual Review of Fluid Mechanics, vol. 17, M. Van Dyke, J.V. Wehausen, and J.L. Lumley, eds., 1985, pp. 487-522.
8. Thompson, J.F.; Warsi, Z.U.A.; and Mastin, C.W.: Numerical Grid Generation. Elsevier Science Publishing Co., 1985.
9. Eiseman, P.R.; and Erlebacher, G.: Grid Generation for the Solution of Partial Differential Equations. NASA CR-178365, 1987.
10. Sengupta, S.; et al., eds.: Numerical Grid Generation in Computational Fluid Mechanics '88. Pineridge Press Limited, 1988.
11. Jameson, A.; and Mavriplis, D.: Finite Volume Solution of the Two-Dimensional Euler Equations on a Regular Triangular Mesh. AIAA Paper 85-0435, Jan. 1985.
12. Mavriplis, D.; and Jameson, A.: Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes. AIAA Paper 87-0353, Jan. 1987.
13. Desideri, J.A.; and Dervieux, A.: Compressible Flow Solvers Using Unstructured Grids. Von Karman Institute Lecture Series 1988-05, Computational Fluid Dynamics, vol. 2, 1988, pp. 1-115.
14. Allmaras, S.R.; and Giles, M.B.: A Second Order Flux Split Scheme for the Unsteady 2-D Euler Equations on Arbitrary Meshes. AIAA Paper 87-1119, 1987.
15. Mavriplis, D.J.: Accurate Multigrid Solution of the Euler Equations on Unstructured and Adaptive Meshes. AIAA/ASME/SIAM/APS First National Fluid Dynamics Congress, Technical Papers, Part 1, AIAA, Washington, D.C., 1988, pp. 419-426.

16. Barth, T.J.; and Jespersen, D.C.: The Design and Application of Upwind Schemes on Unstructured Meshes. AIAA Paper 89-0366, Jan. 1989.

17. Atta, E.H.: Component-Adaptive Grid Embedding. Numerical Grid Generation Techniques, R.E. Smith, ed., NASA CP-2166, 1980, pp. 157-174.

18. Atta, E.H.; and Vadyak, J.: A Grid Overlapping Scheme for Flowfield Computations about Multicomponent Configurations. AIAA J., vol. 21, Sept. 1983, pp. 1271-1277.

19. Steger, J.L.; Dougherty, F.C.; and Benek, J.A.: A Chimera Grid Scheme — Multiple Overset Body-Conforming Mesh System for Finite-Difference Adaptation to Complex Aircraft Configurations. Advances in Grid Generation, K.N. Ghia and U. Ghia, eds., ASME, New York, 1983, pp. 59-69.

20. Dougherty, F.C.; Benek, J.A.; and Steger, J.L.: On Applications of Chimera Grid Schemes to Store Separation. NASA TM-88193, 1985.

21. Benek, J.A.; Donegan, T.L.; and Suh, N.E.: Extended Chimera Grid Embedding Scheme with Application to Viscous Flows. AIAA Paper 87-1126, 1987.

22. Rai, M.M.: A Relaxation Approach to Patched-Grid Calculations with the Euler Equations. J. Comput. Phys., vol. 66, no. 1, 1986, pp. 99-131.

23. Rai, M.M.: A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations. J. Comput. Phys., vol. 62, no. 2, 1986, pp. 472-503.

24. Hessenius, K.; and Rai, M.M.: Three-Dimensional, Conservative, Euler Computations Using Patched Grid Systems and Explicit Methods. AIAA Paper 86-1081, May 1986.

25. Rai, M.M.: Navier-Stokes Simulations of Rotor/Stator Interaction Using Patched and Overlaid Grids. J. Propul. Power, vol. 3, Sept.-Oct. 1987, pp. 387-396.

26. Thomas, P.D.: Composite Three-Dimensional Grids Generated by Elliptic Systems. AIAA J., vol. 20, no. 9, 1982, pp. 1195-1202.

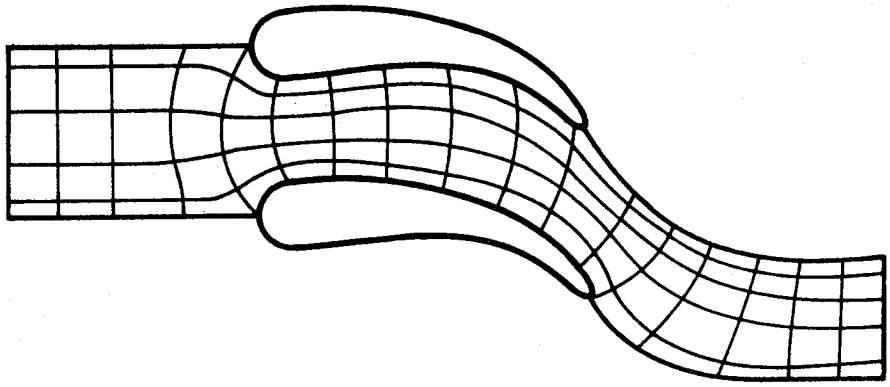
27. Lee, K.D.; Huang, N.J.Y.; and Rubbert, P.E.: Grid Generation for General Three-Dimensional Configurations. Numerical Grid Generation Techniques, NASA CP-2166, R.E. Smith, ed., 1980, pp. 355-366.

28. Rubbert, P.E.; and Lee, K.D.: Patched Coordinate Systems — Numerical Body-Fitted Grid Generation for Surface Boundary Treatment of Aerodynamic Structures. Numerical Grid Generation, J.F. Thompson, ed., Elsevier Publishing Co., 1982, pp. 235-252.

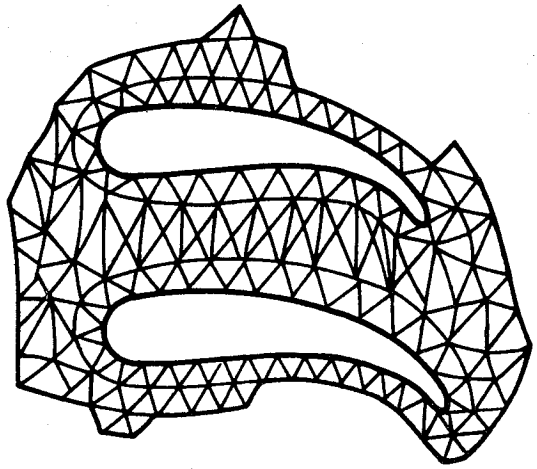
29. Sorenson, R.L.: Grid Generation by Elliptic Partial Differential Equations for a Tri-Element Augmentor-Wing Airfoil. Numerical Grid Generation, J.F. Thompson, ed., Elsevier Publishing Co., 1982, pp. 653-665.

30. Coleman, R.M.: Generation of Boundary-Fitted Coordinate Systems Using Segmented Computational Regions. Numerical Grid Generation, J.F. Thompson, ed., Elsevier Publishing Co., 1982, pp. 633-651.
31. Thompson, J.F.: Composite Grid Generation Code for General 3-D Regions – the EAGLE Code. AIAA J., vol. 26, no. 3, Mar. 1988, pp. 271-272.
32. Gilding, B.H.: A Numerical Grid Generation Technique. Computers and Fluids, vol. 16, no. 1, 1988, pp. 47-58.
33. Coons, S.A.: Surfaces for Computer-Aided Design of Space Forms. Report MAC-TR-41, Contract NONR-4102/01/AF33/600-42859, MIT, Cambridge, Massachusetts, 1967.
34. Gordon, W.J.; and Hall, C.A.: Construction of Curvilinear Coordinate Systems and Applications to Mesh Generation. Int. J. Numer. Methods Eng., vol. 7, 1973, pp. 461-477.
35. Cook, W.A.: Body-Oriented (Natural) Co-ordinates for Generating Three-Dimensional Meshes. Int. J. Numer. Methods Eng., vol. 8, no. 1, 1974, pp. 27-43.
36. Smith, R.E.: Two-Boundary Grid Generation for the Solution of the Three-Dimensional Compressible Navier-Stokes Equations. NASA TM-83123, 1981.
37. Yang, S.-L.; and Shih, T.I-P.: An Algebraic Grid Generation Technique for Time-Varying Two-Dimensional Spatial Domains. Int. J. Numer. Methods Fluids, vol. 6, May 1986, pp. 291-304.
38. Vinokur, M.; and Lombard, C.K.: Algebraic Grid GEneration with Corner Singularity. Advances in Grid Generation, K.N. Ghia and U. Ghia, eds., ASME, New York, 1983, pp. 99-106.
39. Rizzi, A.; and Eriksson, L.E.: Transfinite Mesh Generation and Damped Euler Equation Algorithm for Transonic Flow around Wing-Body Configurations. AIAA Paper 81-0999, 1981.
40. Eriksson, L.E.: Generation of Boundary-Conforming Grids around Wing-Body Configurations Using Transfinite Interpolation. AIAA J., vol. 20, no. 10, Oct. 1982, pp. 1313-1320.
41. Eiseman, P.R.: A Multi-Surface Method of Coordinate Generation. J. Comput. Phys., vol. 33, Oct. 1979, pp. 118-150.
42. Eiseman, P.R.: Coordinate Generation with Precise Controls over Mesh Properties. J. Comput. Phys., vol. 47, Sept. 1982, pp. 331-351.
43. Eiseman, P.R.: High Level Continuity for Coordinate Generation with Precise Controls. J. Comput. Phys., vol. 47, Sept. 1982, pp. 352-374.
44. Roberts, G.O.: Computational Meshes for Boundary Layer Problems. Lecture Notes in Physics, vol. 8, 1971, pp. 171-177.

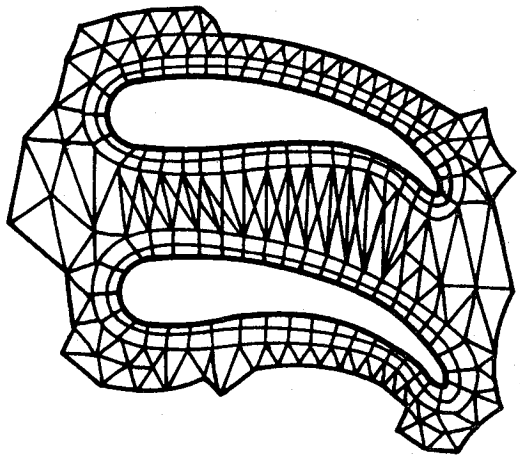
45. Vinokur, M.: On One-Dimensional Stretching Functions for Finite-Difference Calculations – Computational Fluid Dynamics. J. Comput. Phys., vol. 50, May 1983, pp. 215-234.
46. Birkhoff, G.; and Garabedian, H.: Smooth Surface Interpolation. J. Mathe. Phys., vol. 39, 1960, pp. 258-268.
47. Schweikert, D.G.: An Interpolation Curve Using a Spline in Tension. J. Math. Phys., vol. 45, 1966, pp. 312-317.
48. De Boor, C.: Bicubic Spline Interpolation. J. Math. Phys., vol. 41, no. 3, 1962, pp. 212-218.
49. Ferguson, J.: Multivariable Curve Interpolation. Assoc. Comput. Mach., vol. 11, Apr. 1964, pp. 221-228.
50. Spath, H.: Algorithm 16: Two-Dimensional Exponential Splines. Computing, vol. 4, 1969, pp. 225-233.
51. Shih, T. I-P.: Finite-Difference Methods in Computational Fluid Dynamics. Prentice-Hall, to be published.
52. Thomas, P.D.; and Lombard, C.K.: Geometric Conservation Law and Its Application to Flow Computations on Moving Grids. AIAA J., vol. 17, no. 10, 1979, pp. 1030-1037.
53. Hindman, R.G.: Generalized Coordinate Forms of Governing Fluid Equations and Associated Geometrically Induced Errors. AIAA J., vol. 20, no. 10, 1982, pp. 1359-1367.
54. Steger, J.L.: On Application of Body Conforming Curvilinear Grids for Finite Difference Solution of External Flow. Numerical Grid Generation, J.F. Thompson, ed., Elsevier Science Publishing Co., 1982, pp. 295-315.
55. Cheney, W.; and Kincaid, D.: Numerical Mathematics and Computing. Brooks/Cole Publishing Co., Monterey, CA, 1980, pp. 151-155.
56. Ferziger, J.H.: Numerical Methods for Engineering Application. John Wiley and Sons, 1981, pp. 12-23.
57. Chapra, S.C.; and Canale, R.P.: Numerical Methods for Engineers with Personal Computer Applications. McGraw-Hill Book Co., 1985, pp. 302-306.
58. Ralston, A.; and Rabinowitz, P.: A First Course in Numerical Analysis, Second ed. McGraw-Hill Book Co., 1978.
59. Dahlquist, G.; and Bjorck, A.: Numerical Methods. Prentice-Hall, 1974.
60. Anderson, D.A.; Tannehill, J.C.; and Pletcher, R.H.: Computational Fluid Mechanics and Heat Transfer. Hemisphere Publishing Corp., Washington, D.C., 1984.



(A)



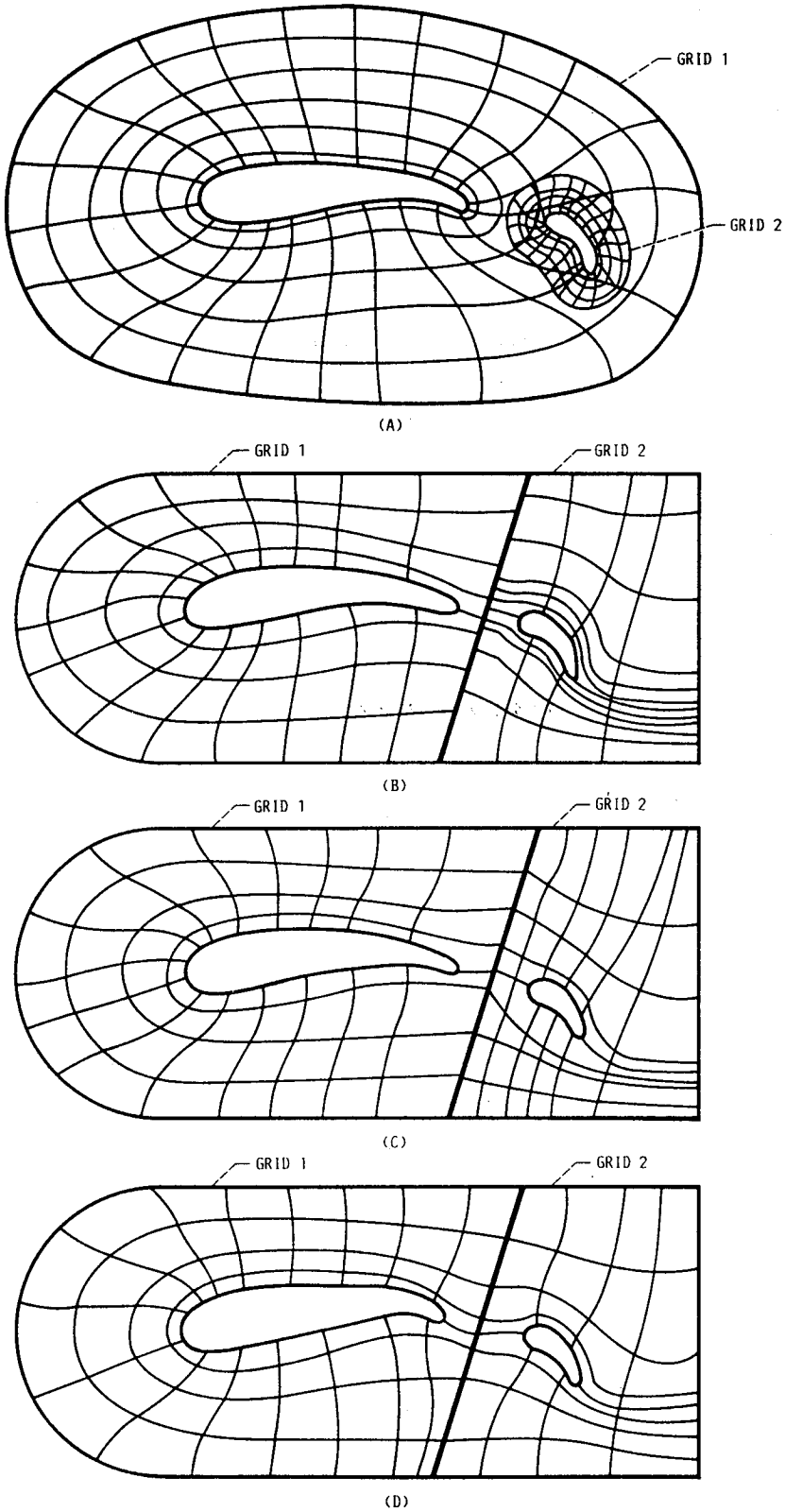
(B)



(C)

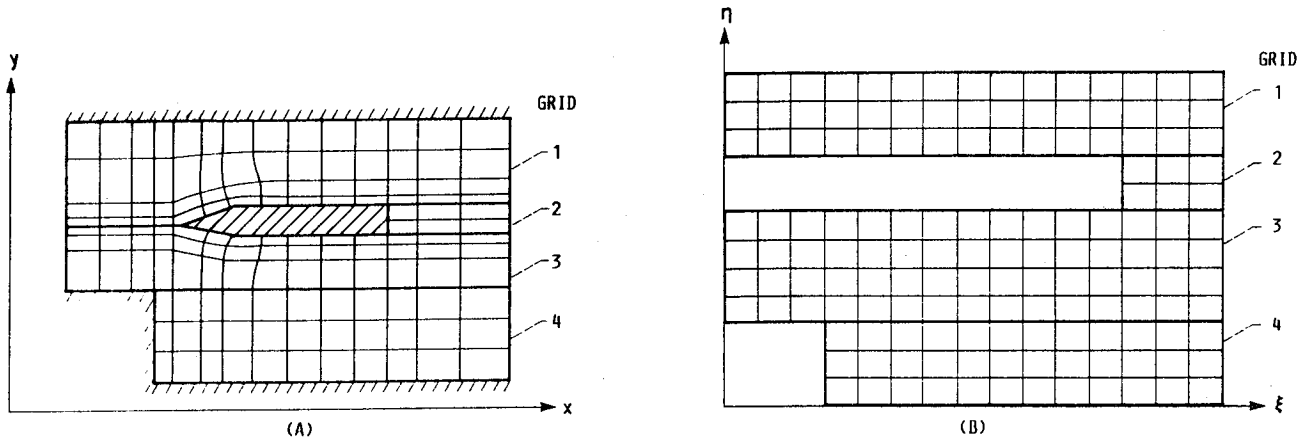
(a) STRUCTURED.
(b) UNSTRUCTURED.
(c) MIXED (PARTLY STRUCTURED, PARTLY UNSTRUCTURED).

FIGURE 1-1. - TYPES OF GRID SYSTEMS.



(a) COMPLETELY DISCONTINUOUS.
(b) PARTIALLY DISCONTINUOUS.
(c) PARTIALLY CONTINUOUS.
(d) PARTIALLY OR COMPLETELY CONTINUOUS.

FIGURE 1-2. - TYPES OF COMPOSITE GRIDS.



(a) IN SPATIAL DOMAIN (x-y COORDINATE SYSTEM).
(b) IN TRANSFORMED DOMAIN (ξ-η COORDINATE SYSTEM).

FIGURE 1-3. - CONTINUOUS COMPOSITE GRID SYSTEM MADE UP OF FOUR SINGLE GRIDS.

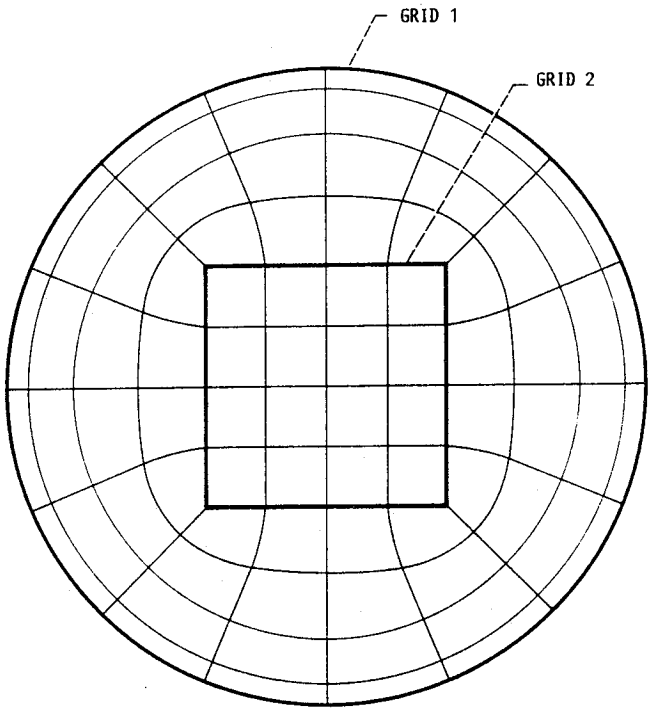


FIGURE 1-4. - PARTIALLY DISCONTINUOUS COMPOSITE GRID THAT APPEARS LIKE A PARTIALLY OR COMPLETELY CONTINUOUS COMPOSITE GRID.

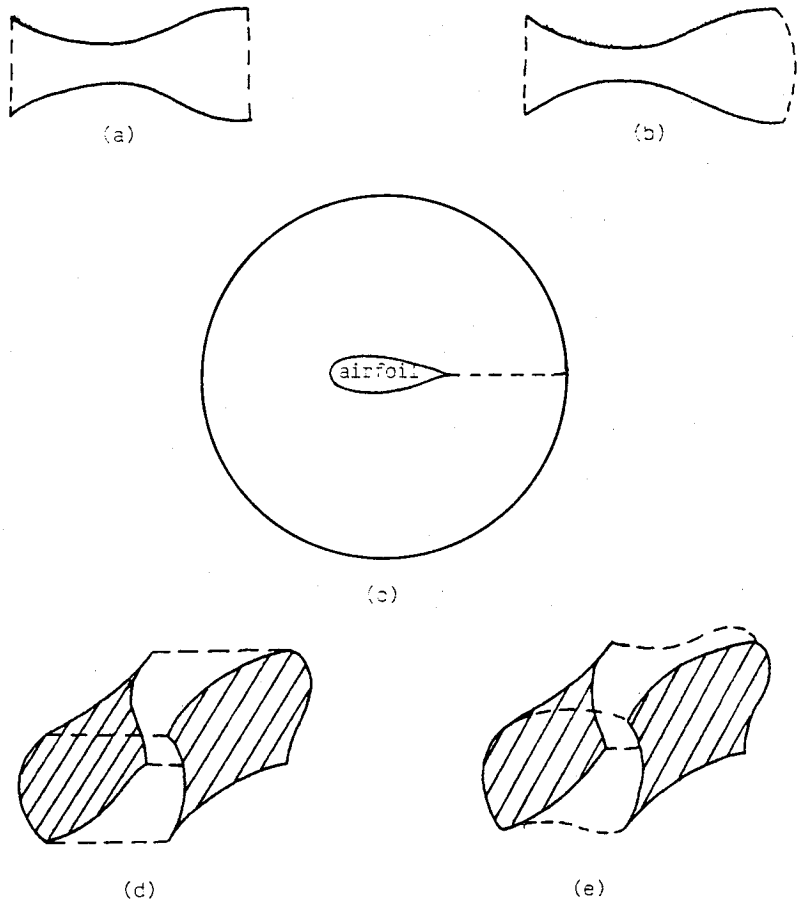
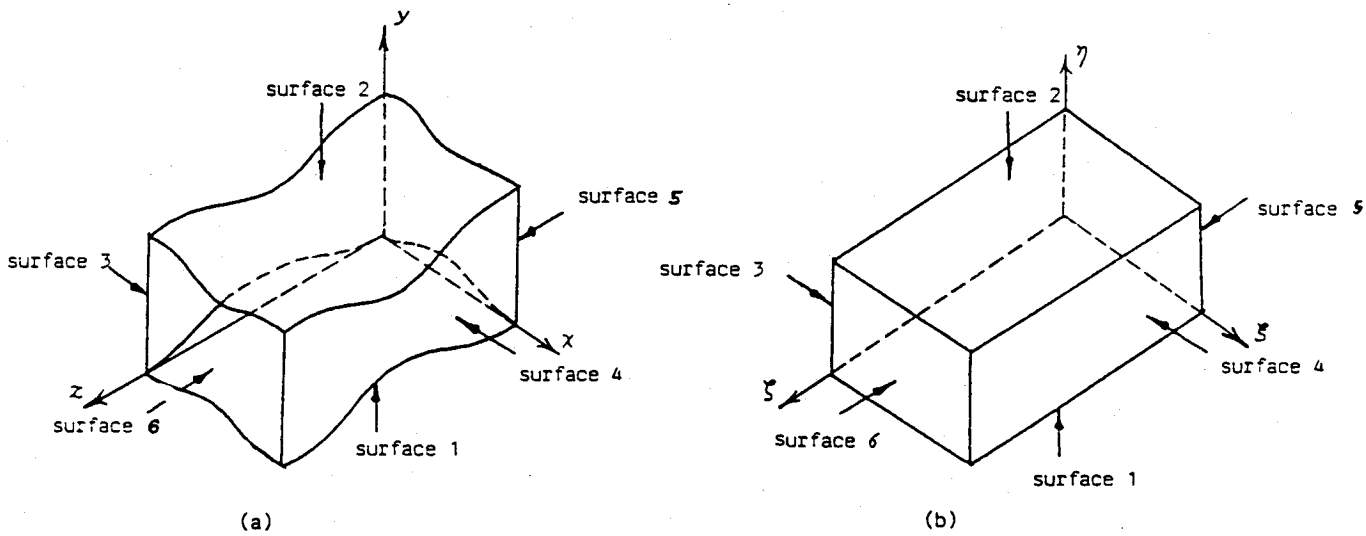
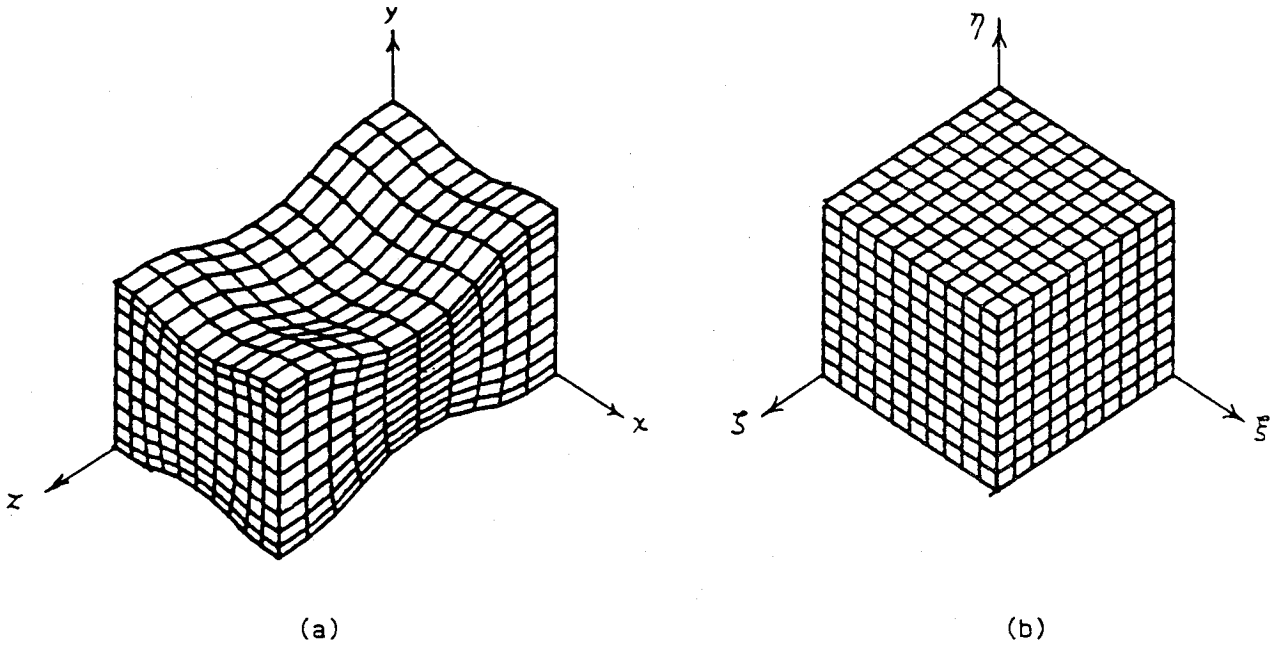


FIGURE 2-1. - EXAMPLES OF SPATIAL DOMAINS THAT ARE WELL-SUITED FOR THE TWO-BOUNDARY METHOD. THE TWO BOUNDARIES SUGGESTED ARE SHOWN AS SOLID LINES IN THE 2-D CASE AND SHADED SURFACES IN THE 3-D CASE.



(a) IN x-y-z-t COORDINATE SYSTEM.
(b) IN ξ-η-ζ-τ COORDINATE SYSTEM.

FIGURE 2-2. - SPATIAL DOMAIN.



(a) IN SPATIAL DOMAIN.
(b) IN TRANSFORMED DOMAIN.

FIGURE 2-3. - GRID SYSTEMS.

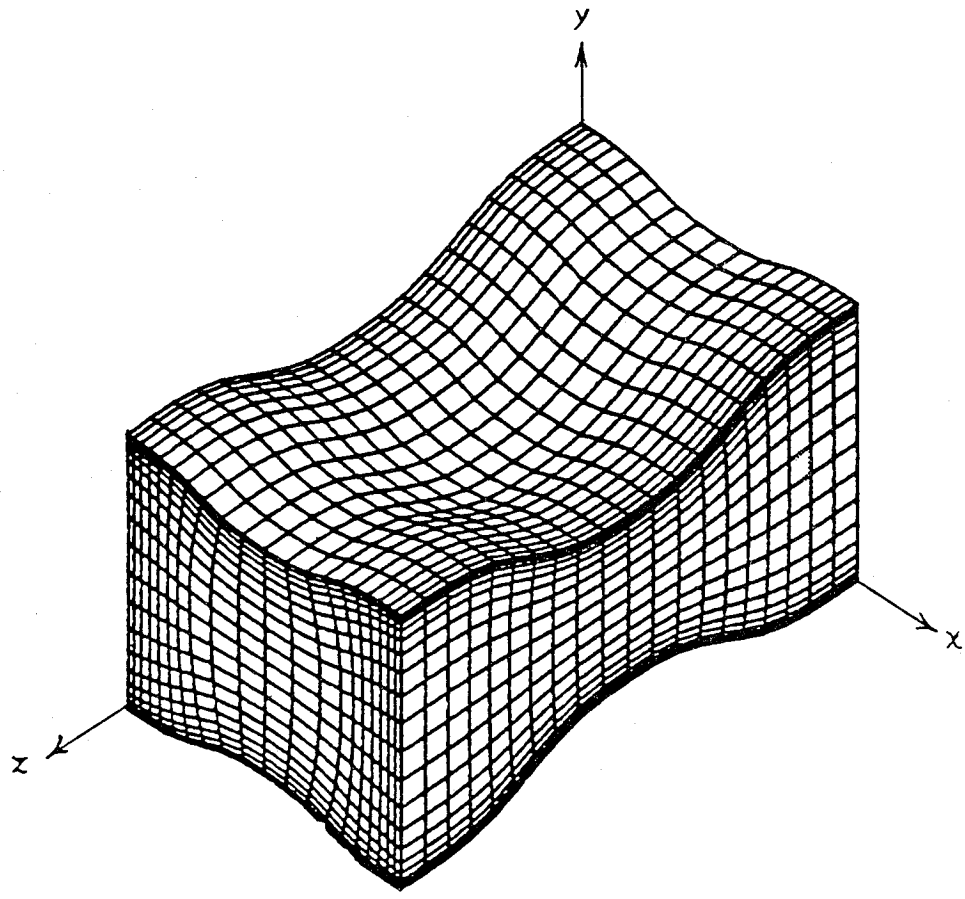
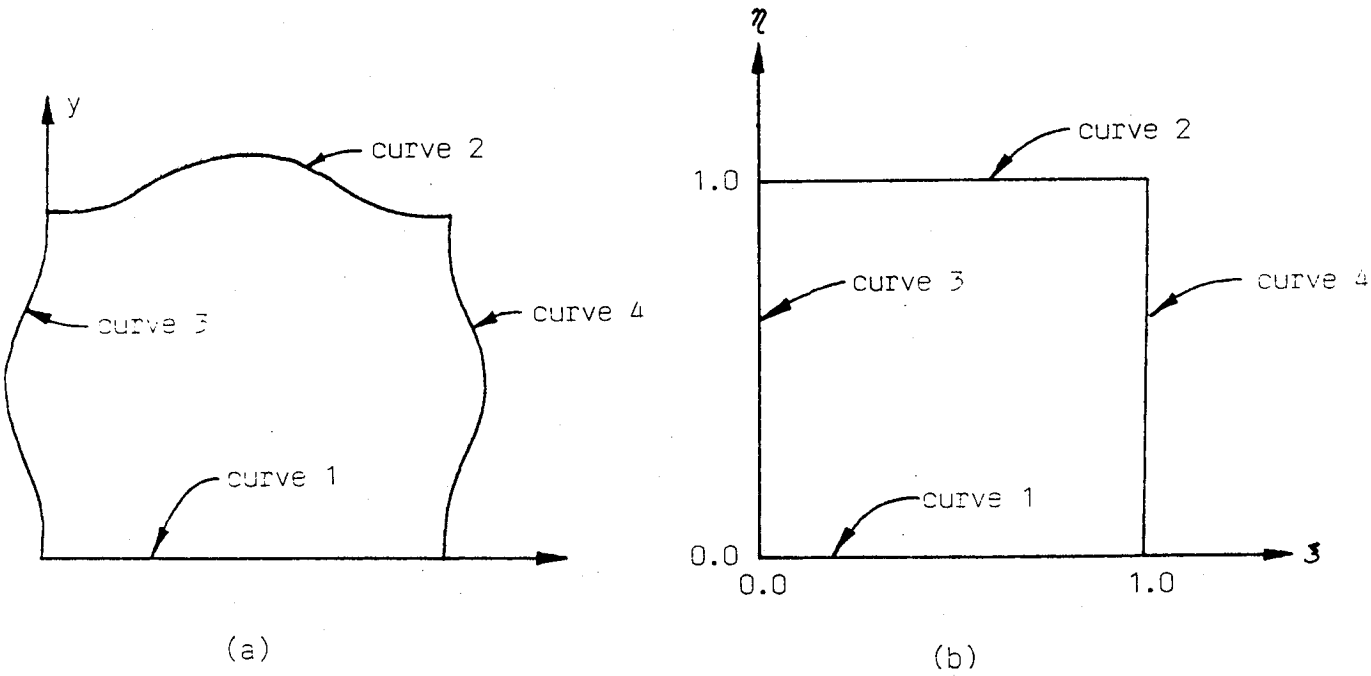


FIGURE 2-4. - GRID SYSTEM IN THE SPATIAL DOMAIN AFTER STRETCHING. GRID IS COMPUTER GENERATED.



(a) IN x-y COORDINATE SYSTEM.
(b) IN ξ-η COORDINATE SYSTEM.

FIGURE 2-5. - SPATIAL DOMAIN.

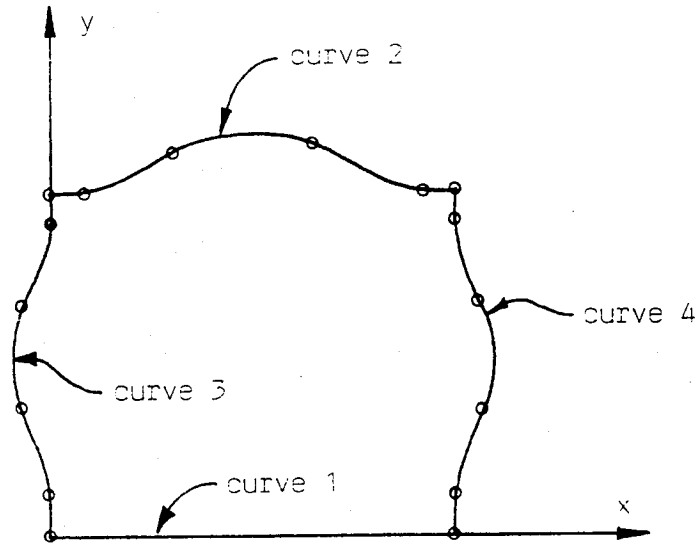


FIGURE 2-6. - APPROXIMATIONS OF CURVES 1, 2, 3, AND 4 OBTAINED BY USING PARAMETRIC TENSION SPLINE INTERPOLATIONS. THE CURVES ARE COMPUTER GENERATED AND NODAL POINTS ARE DENOTED BY o.

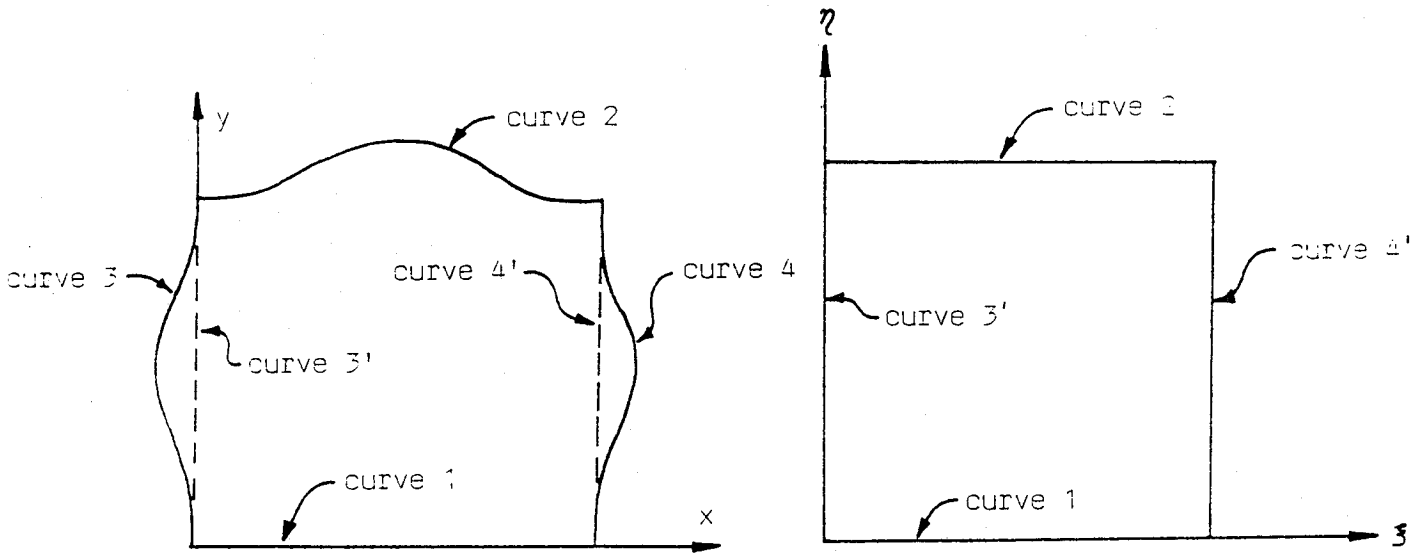
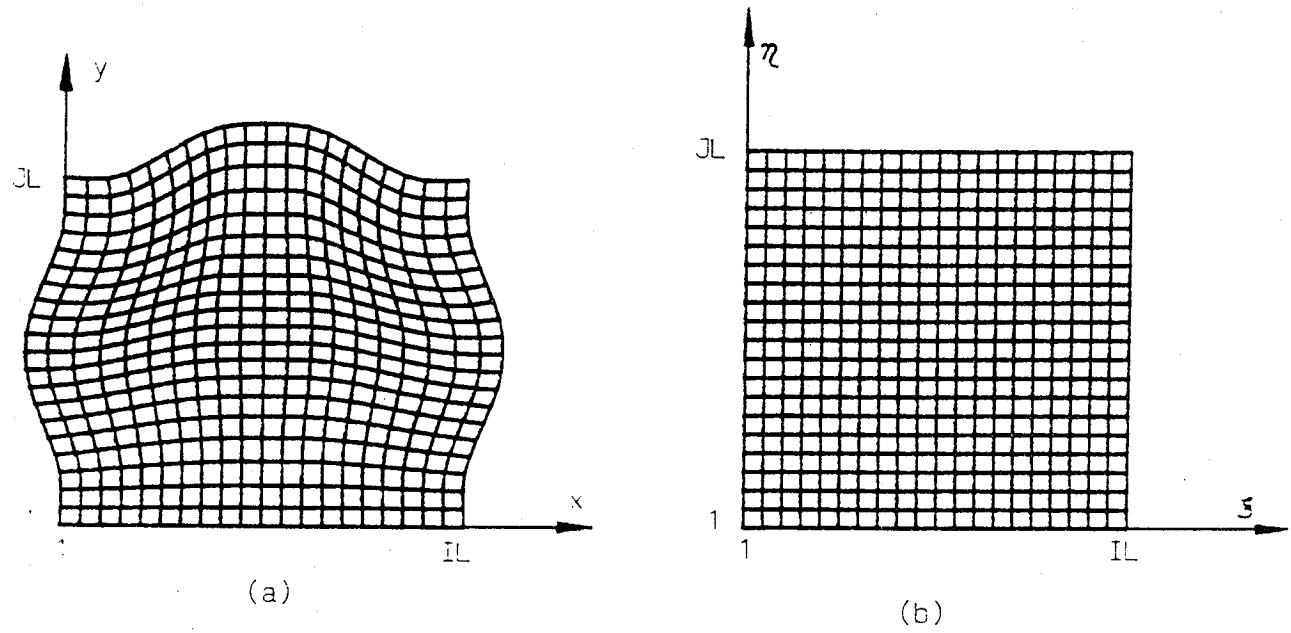
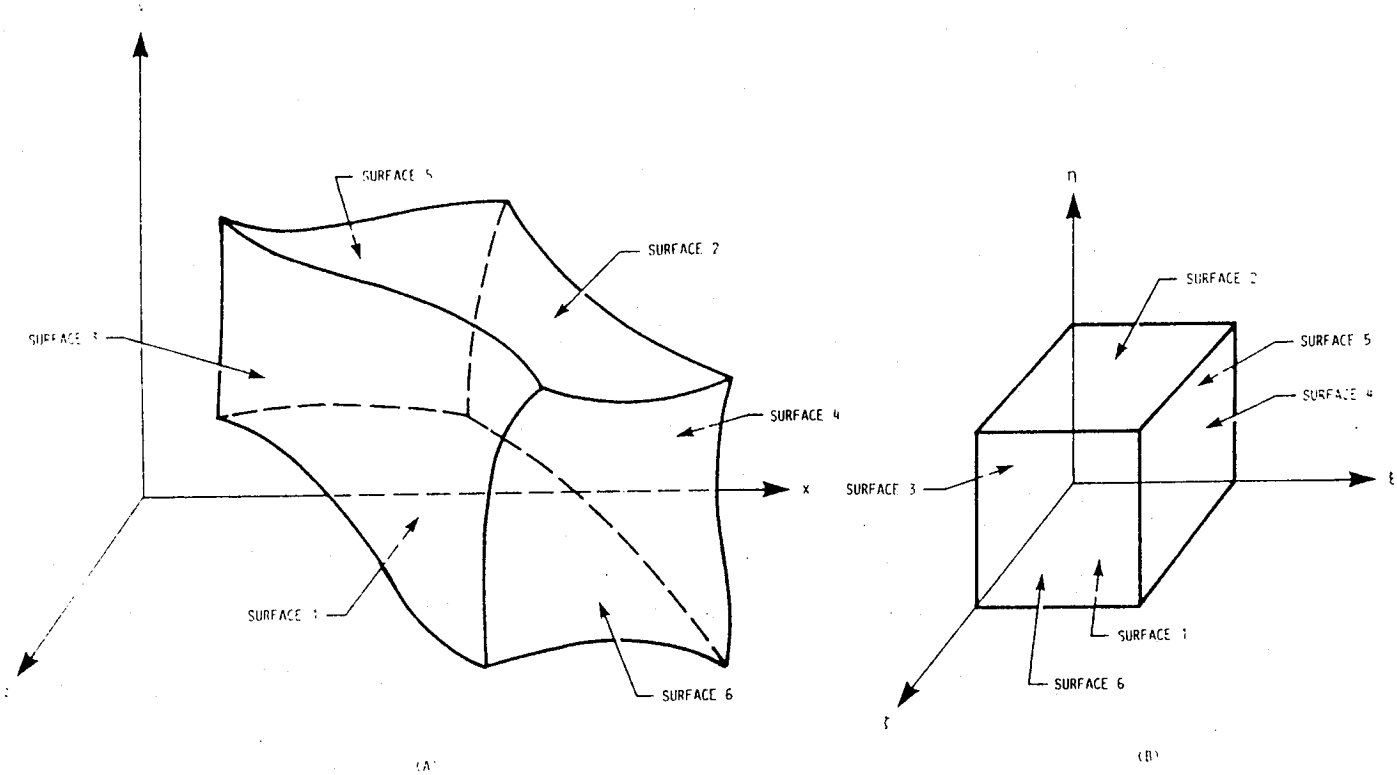


FIGURE 2-7. - MAPPING BETWEEN THE SPATIAL AND TRANSFORMED DOMAINS OBTAINED BY USING EQUATION (2.21). NOTE THAT CURVES 3' AND 4' IN THE SPATIAL DOMAIN ARE MAPPED TO COORDINATE LINES $\xi = 0$ AND $\xi = 1$ IN THE TRANSFORMED DOMAIN, RESPECTIVELY.



(a) IN SPATIAL DOMAIN.
(b) IN TRANSFORMED DOMAIN.

FIGURE 2-8. - GRID SYSTEM.



(a) SPATIAL DOMAIN IN x-y-z-t COORDINATE SYSTEM.
(b) TRANSFORMED DOMAIN IN ξ-η-ζ-τ COORDINATE SYSTEM.

FIGURE 2-9. - SPATIAL DOMAIN.

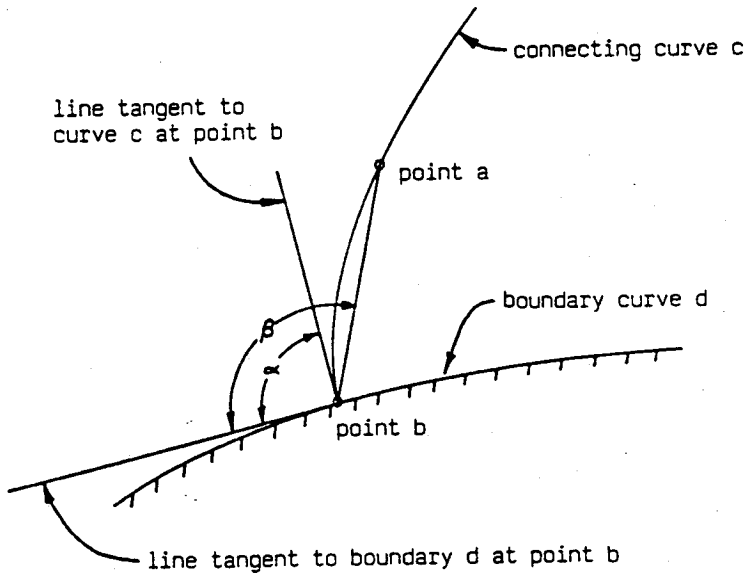


FIGURE 2-10. - DIAGRAM SHOWING HOW ORTHOGONALITY OF A CONNECTING CURVE AT A BOUNDARY DOES NOT GUARANTEE ORTHOGONALITY AFTER THE CONNECTING CURVE HAS BEEN REPLACED BY GRID POINTS.

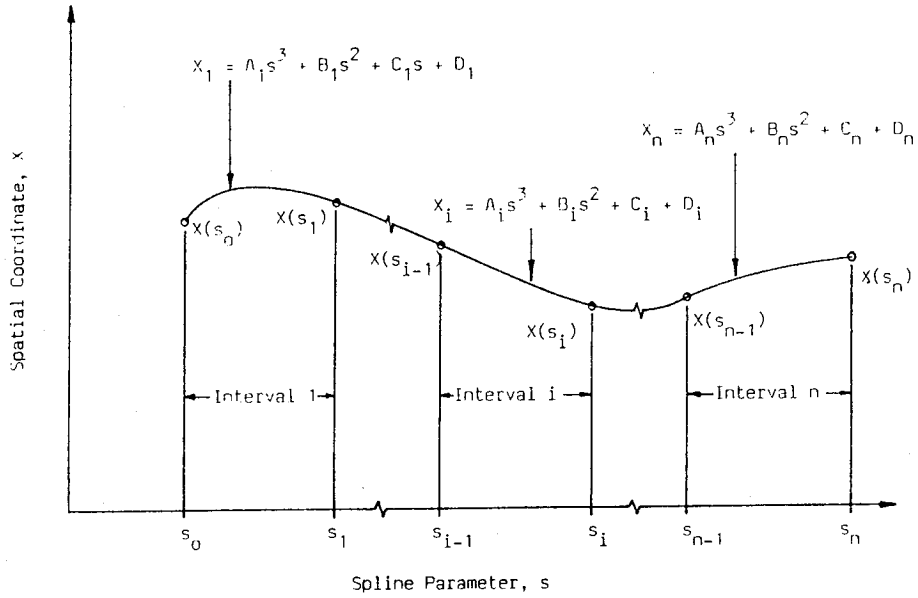
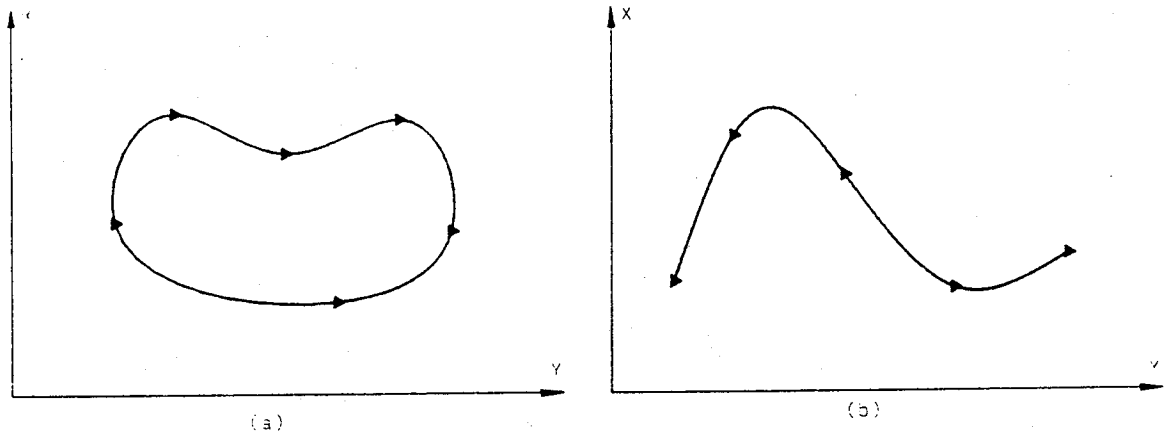
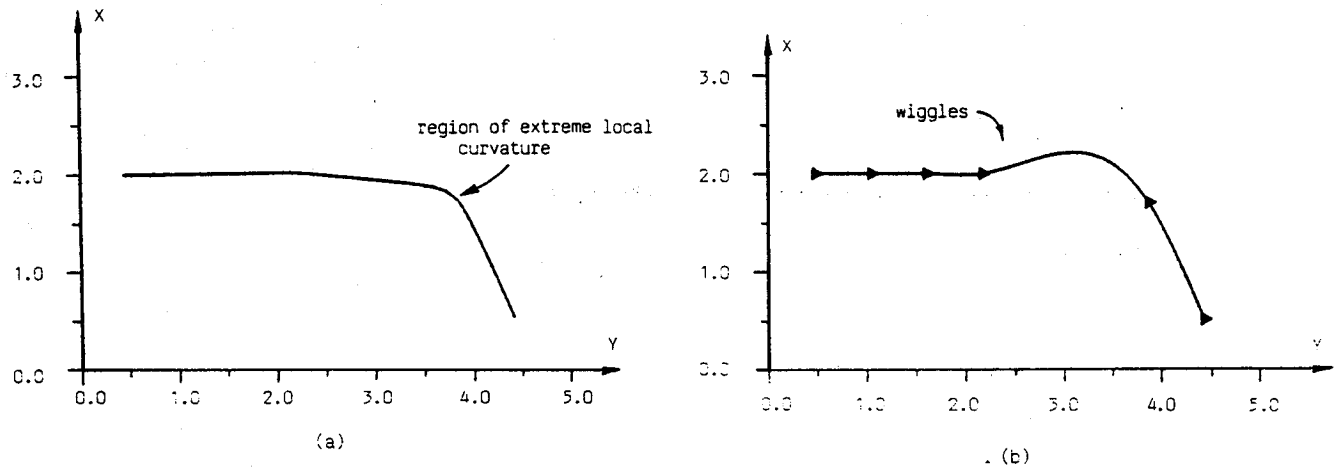


FIGURE 3-1. - NOTATION USED IN DERIVING PARAMETRIC CUBIC SPLINES. NODAL POINTS ARE DENOTED BY o.



(a) CYCLIC END CONDITIONS.
(b) NATURAL END CONDITIONS.

FIGURE 3-2. - EXAMPLES OF CURVES GENERATED BY USING PARAMETRIC CUBIC SPLINE INTERPOLATION. THE CURVES ARE COMPUTER GENERATED AND NODAL POINTS ARE DENOTED BY ►.



(a) CURVE BEING APPROXIMATED.
(b) CUBIC SPLINE APPROXIMATION.

FIGURE 3-3. - CASE WHERE A CUBIC SPLINE POSSESSES UNDESIRABLE WIGGLES. CURVE IN (b) IS COMPUTER GENERATED AND NODAL POINTS ARE DENOTED BY ►.

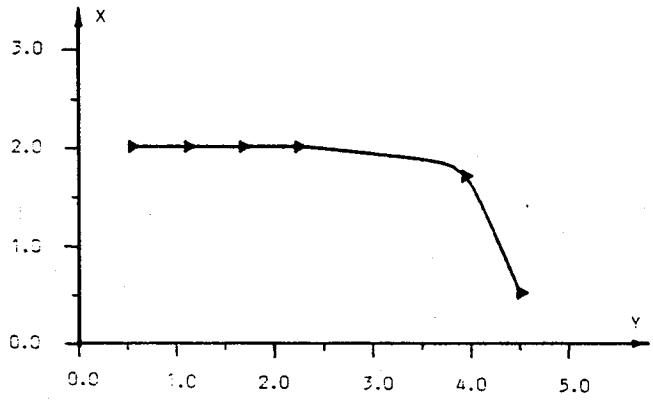


FIGURE 3-4. - A PARAMETRIC TENSION SPLINE WITH σ = 10 FOR THE SAME SET OF NODAL POINTS AS THE CUBIC SPLINE OF FIGURE 3-3(b). NOTE THAT THE WIGGLES PRESENT IN THE CUBIC SPLINE SHOWN IN FIGURE 3-3(b) HAVE BEEN REMOVED. THE CURVE IS COMPUTER GENERATED AND NODAL POINTS ARE DENOTED BY ►.

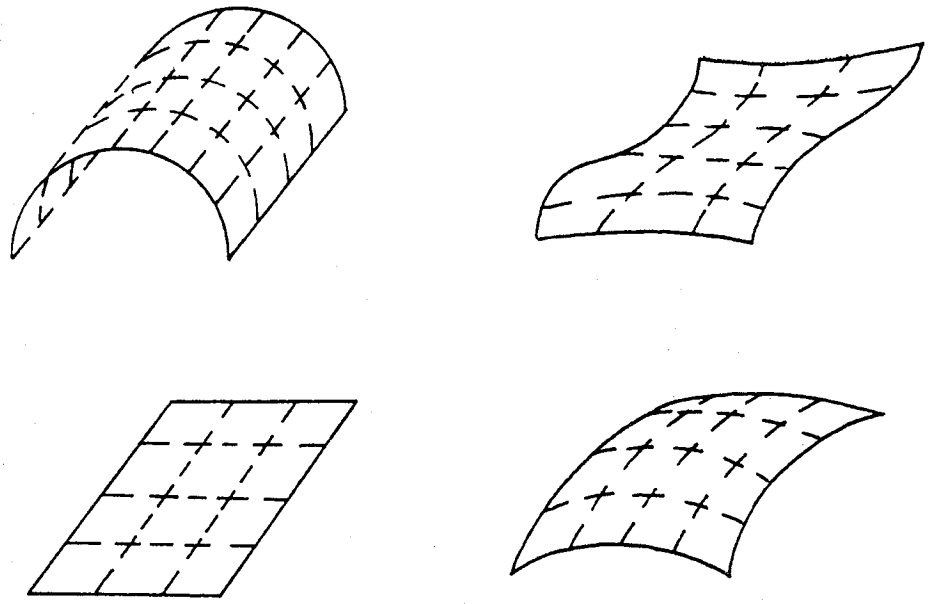


FIGURE 3-5. - EXAMPLES OF SURFACES WHOSE INTERIOR REGIONS ARE "SIMILAR" TO THEIR BOUNDARY CURVES. BOUNDARY CURVES ARE SHOWN AS SOLID LINES, WHILE INTERIOR CURVES ARE SHOWN AS DASHED LINES.

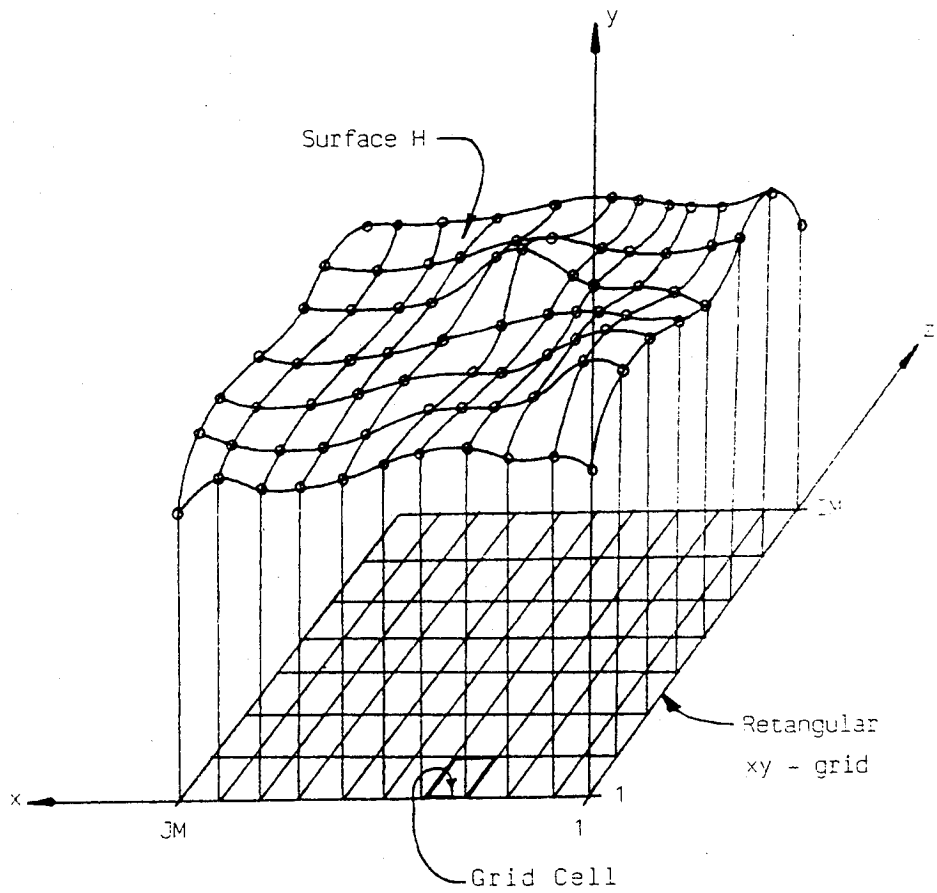


FIGURE 3-6. - POINTS ON SURFACE H GIVEN ON A RECTANGULAR x-y GRID. NODAL POINTS ARE DENOTED BY o.

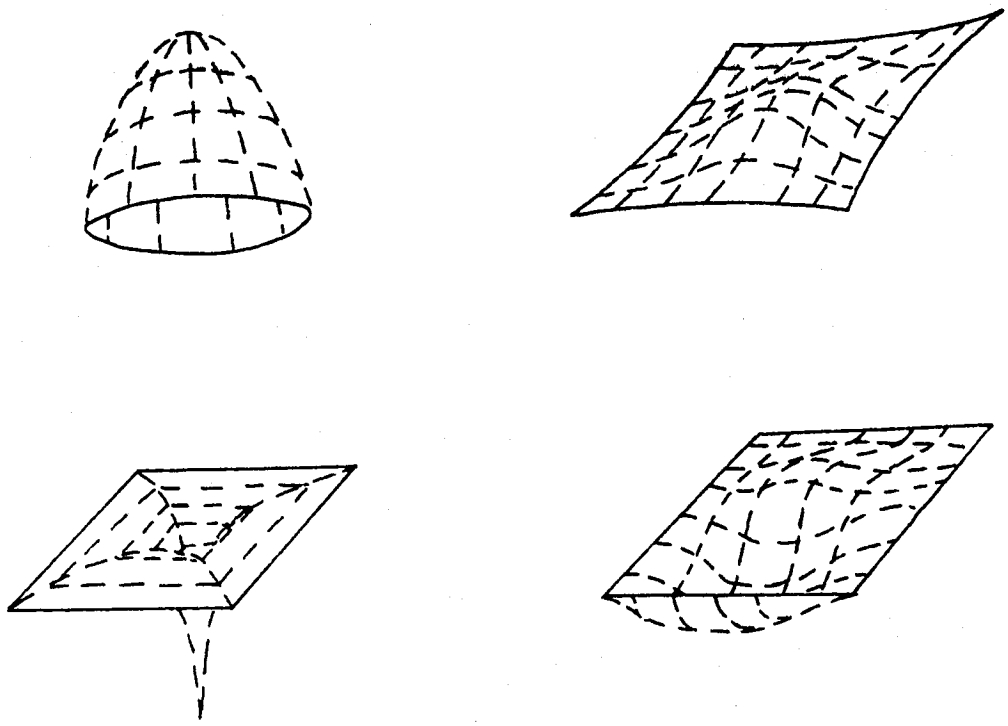


FIGURE 3-7. - EXAMPLES OF SURFACES WHOSE INTERIOR REGIONS DIFFER GREATLY FROM THEIR BOUNDARY CURVES. BOUNDARY CURVES ARE SHOWN AS SOLID LINES, WHILE INTERIOR CURVES ARE SHOWN AS DASHED LINES.

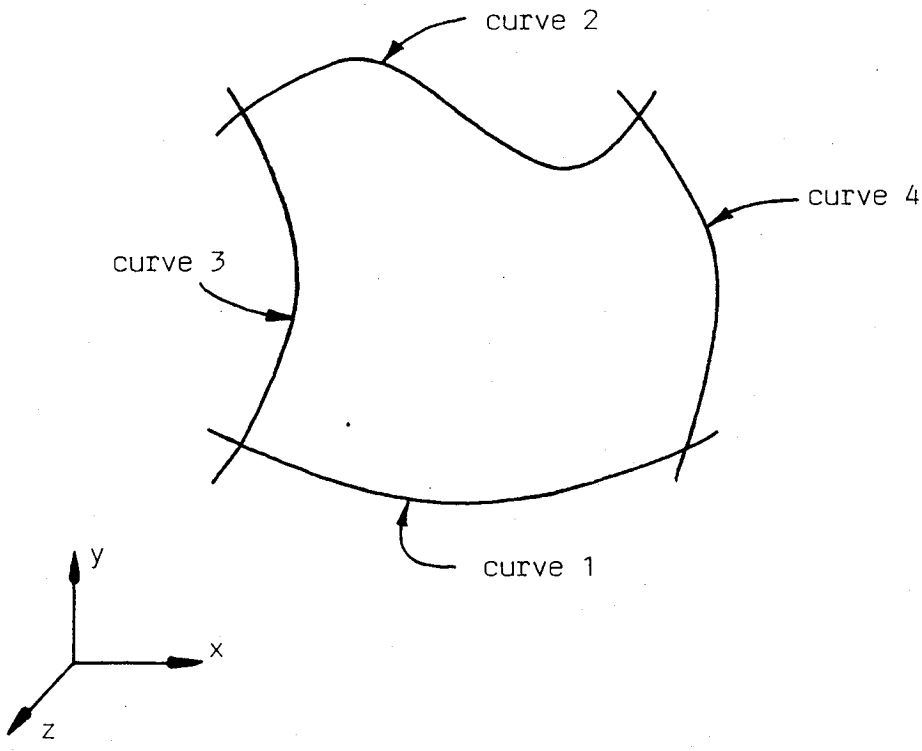


FIGURE 3-8. - FOUR TWISTED CURVES WHICH INTERSECT TO FORM A CURVILINEAR "QUADRILATERAL."

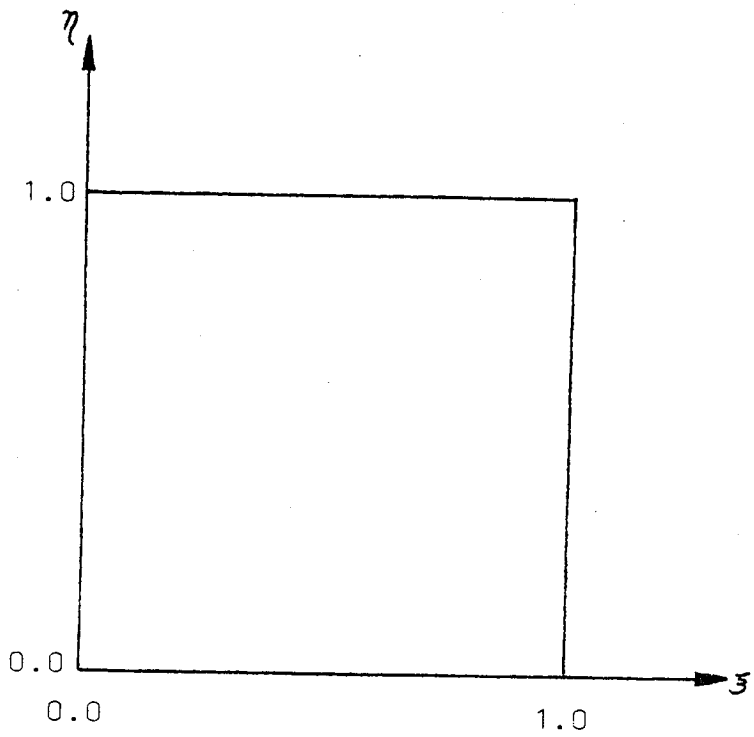
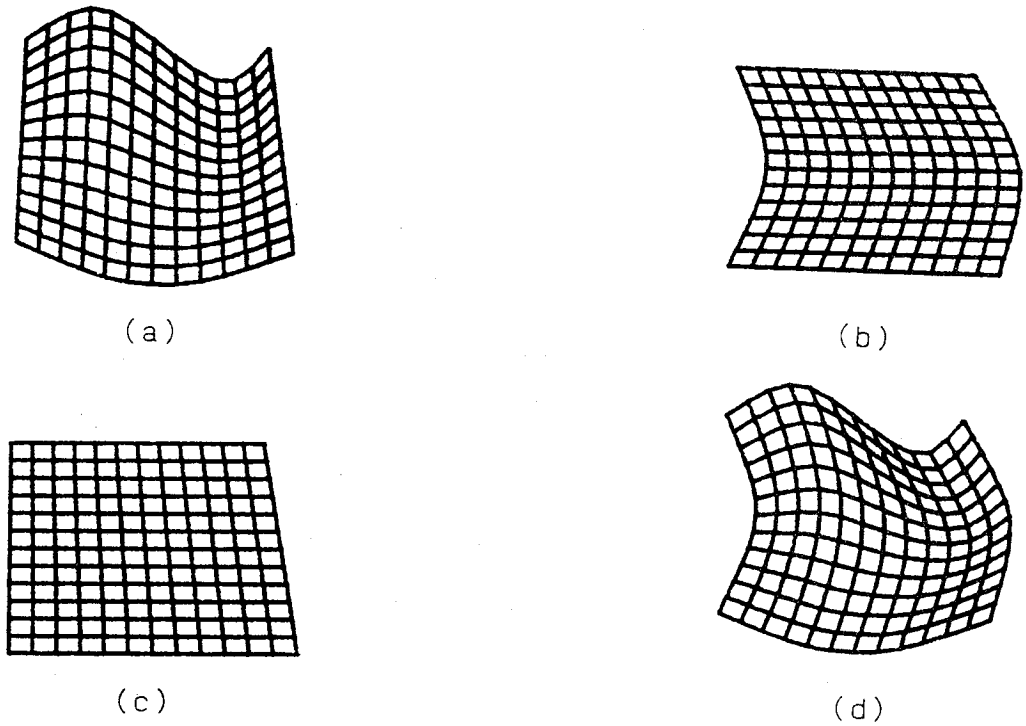


FIGURE 3-9. - A COORDINATE PLANE IN THE ξ-η-ζ-τ COORDINATE SYSTEM AT $\zeta = 0$.



(a) FIRST.
(b) SECOND.
(c) THIRD.
(d) FOURTH.

FIGURE 3-10. - FOUR STAGES OF TRANSFINITE BILINEAR INTERPOLATION.

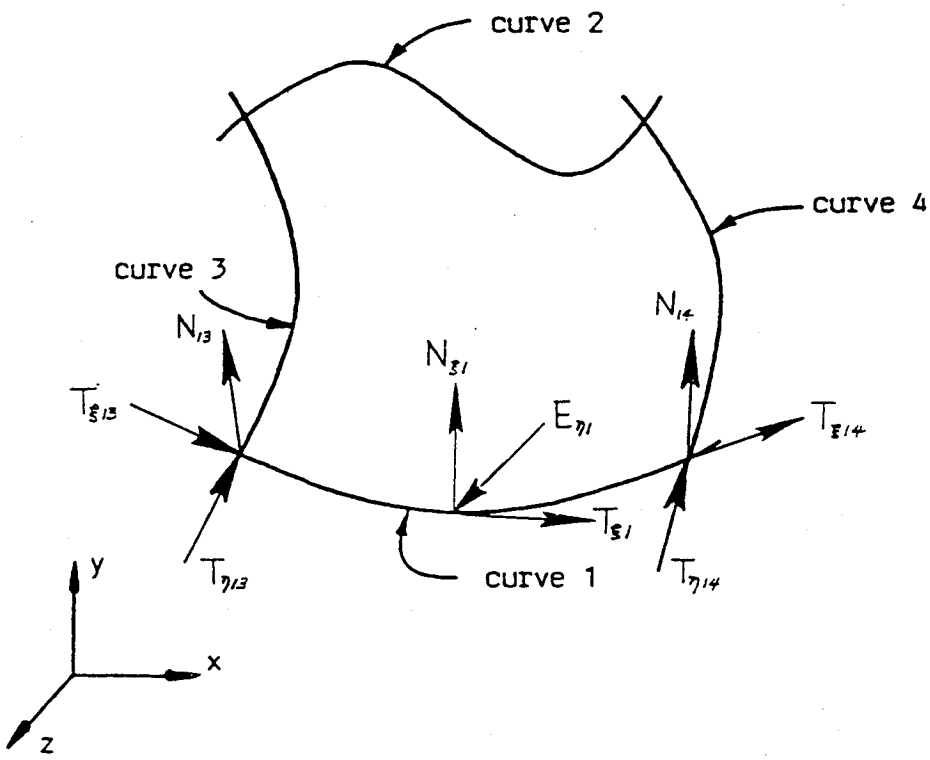


FIGURE 3.11. - DEFINITIONS OF VECTORS TANGENT AND ORTHOGONAL TO CURVE 1.

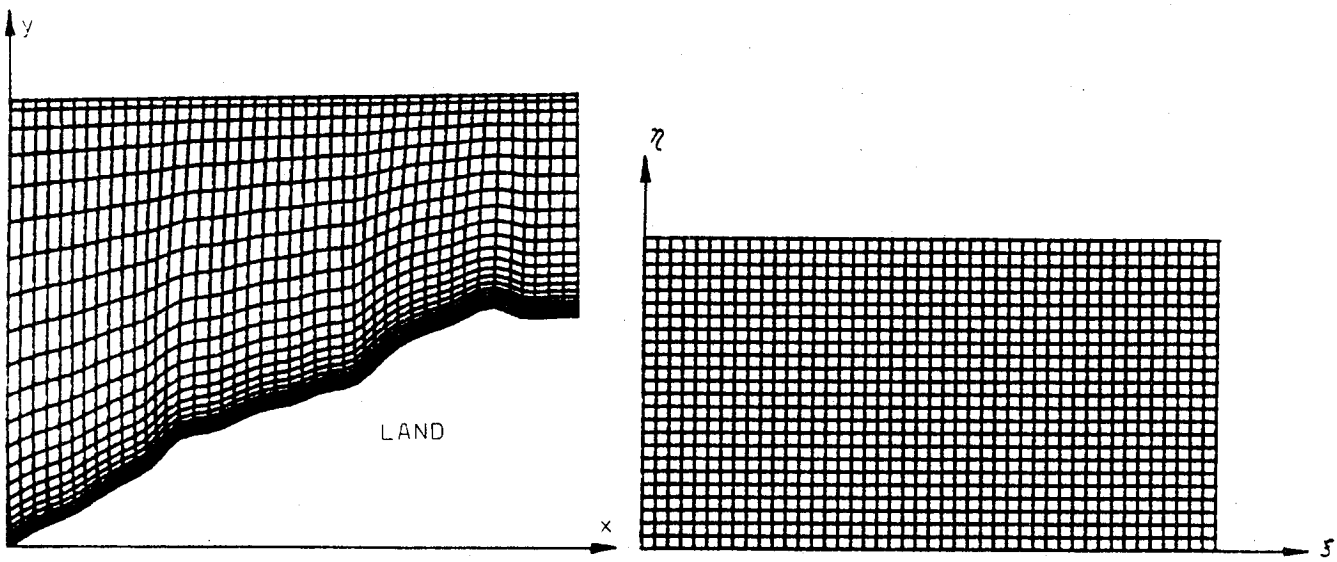


FIGURE 4.1. GRID SYSTEM GENERATED FOR AN IRREGULAR COASTLINE TOPOLOGY BY USING THE TWO-BOUNDARY METHOD. GRID IS COMPUTER GENERATED.

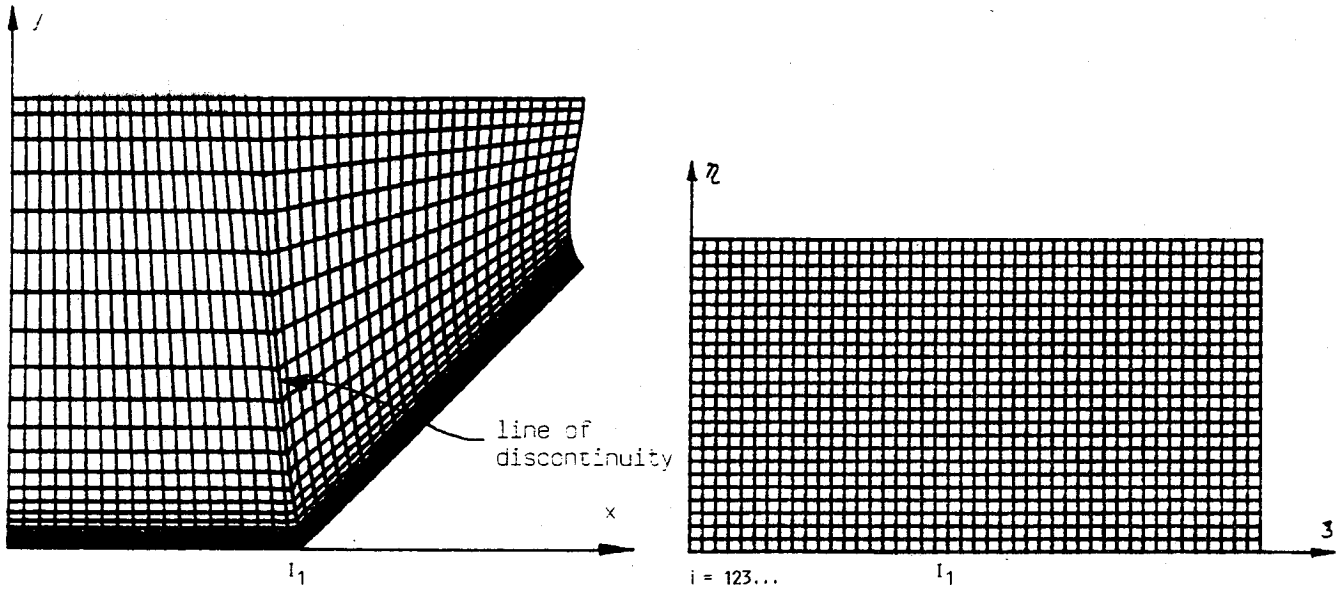


FIGURE 4-2. - GRID SYSTEM GENERATED ABOVE A SHARP BEND BY USING THE TWO-BOUNDARY METHOD. GRID IS COMPUTER GENERATED.

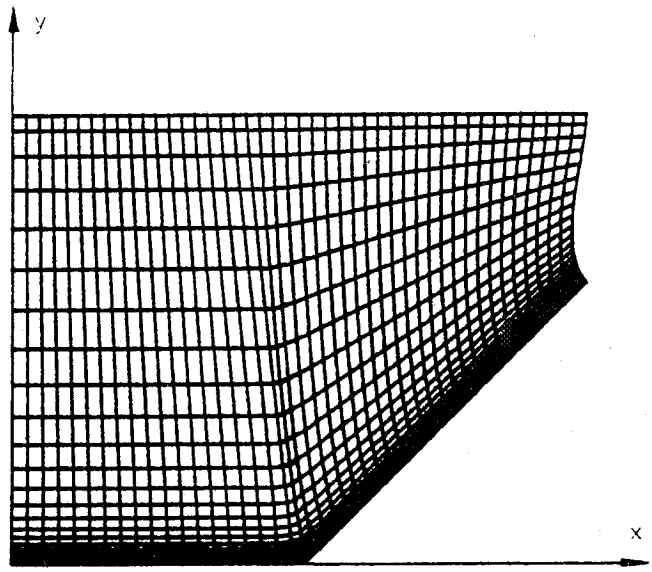


FIGURE 4-3. - GRID SYSTEM OF FIGURE 4-2 AFTER SMOOTHING. GRID IS COMPUTER GENERATED.

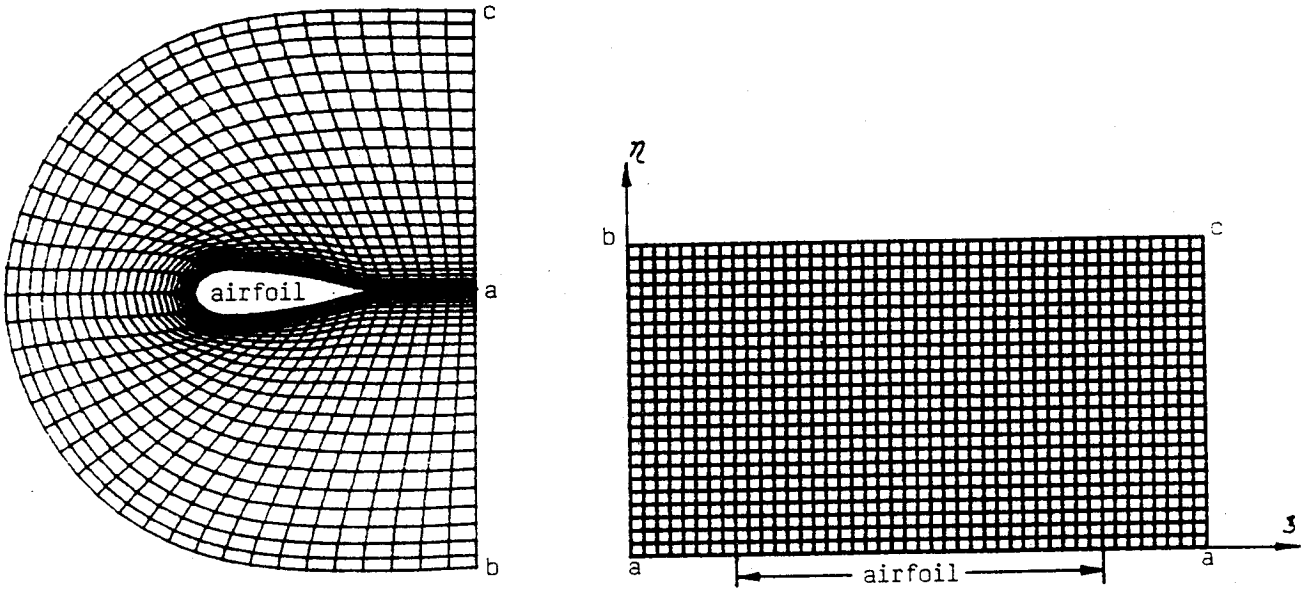


FIGURE 4-4. - A C-GRID GENERATED ABOUT AN AIRFOIL BY USING THE TWO-BOUNDARY METHOD. GRID IS COMPUTER GENERATED.

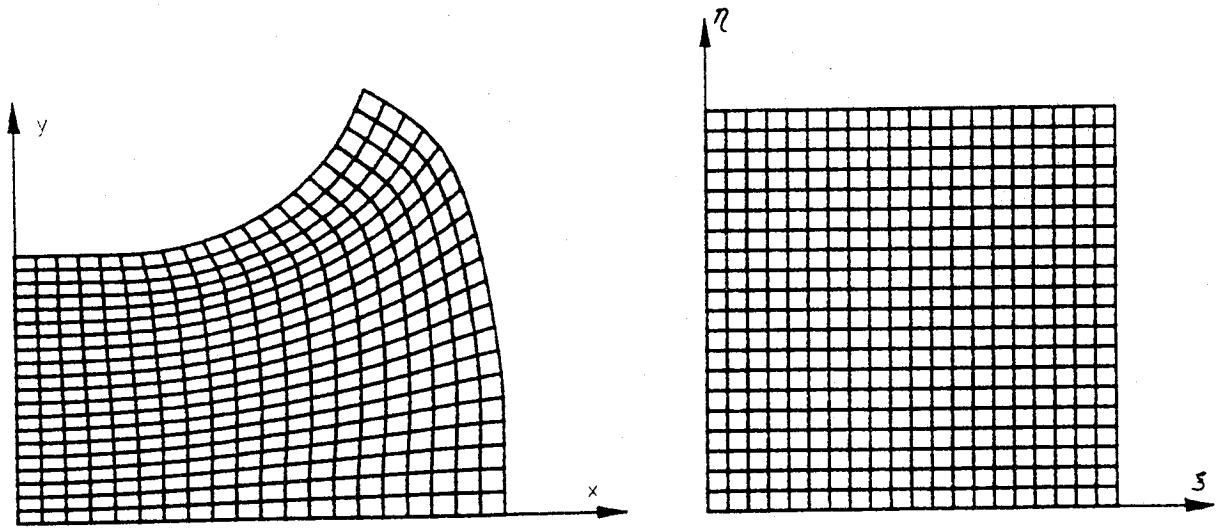


FIGURE 4-5. - GRID GENERATED FOR AN IRREGULARLY SHAPED, FOUR-SIDED REGION BY USING THE FOUR-BOUNDARY METHOD. GRID IS COMPUTER GENERATED.

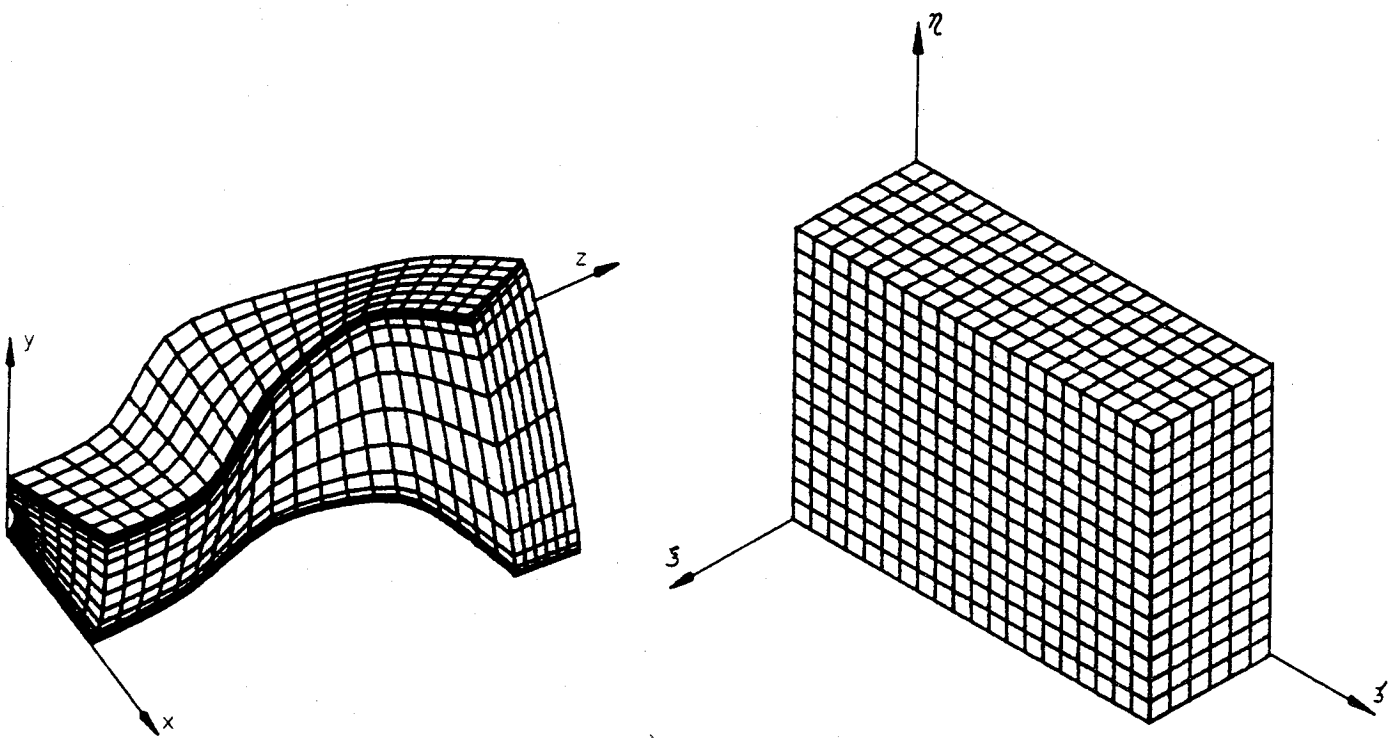


FIGURE 4-6. - GRID GENERATED BETWEEN THE BLADES OF A RADIAL TURBINE BY USING THE TWO-BOUNDARY METHOD. GRID IS COMPUTER GENERATED.

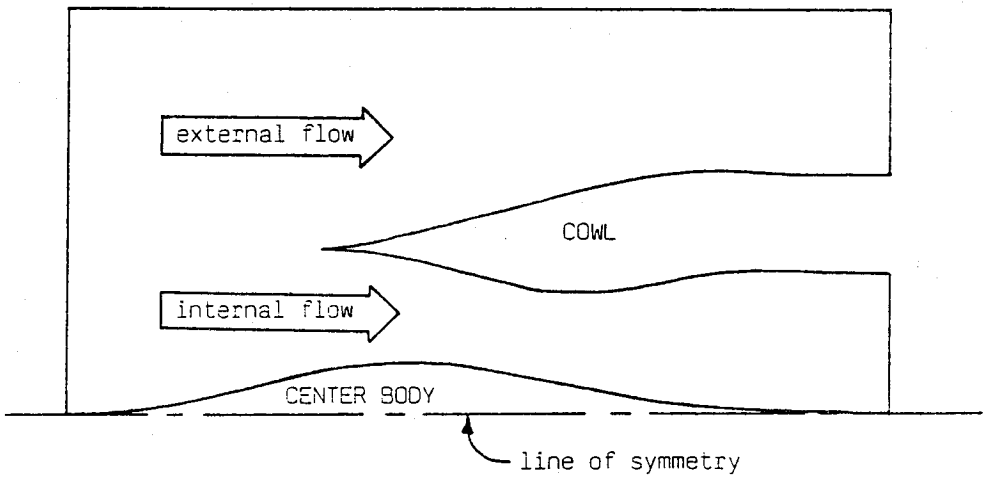
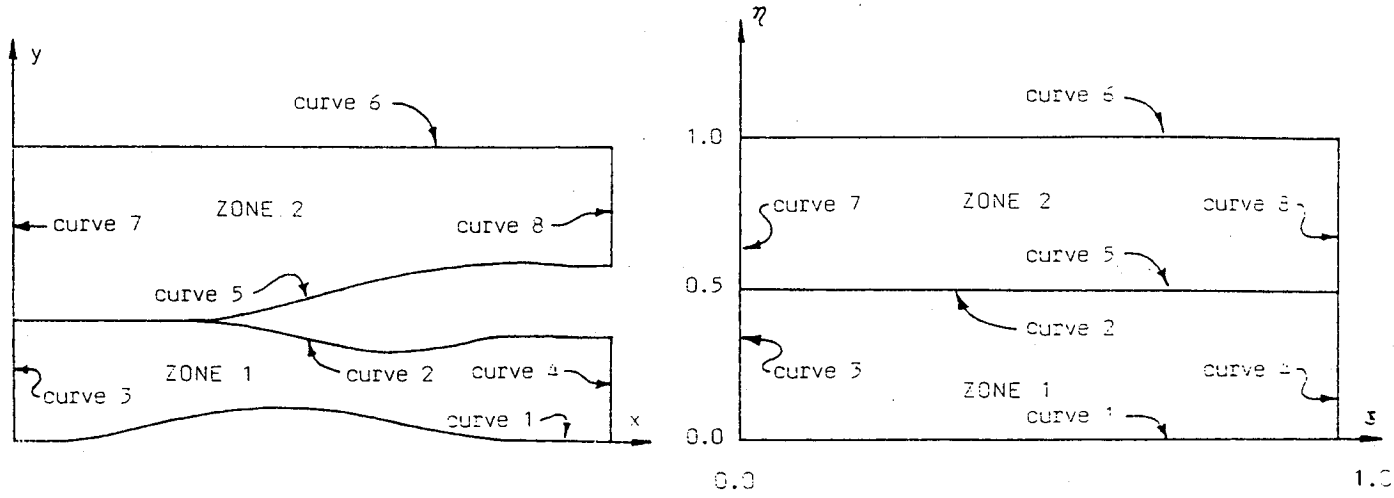


FIGURE 4-7. - TWO-DIMENSIONAL INLET OF TURBOJET ENGINE.

C-2



(a) IN x-y COORDINATE SYSTEM.
(b) IN ξ-η COORDINATE SYSTEM.

FIGURE 4-8. - PARTITIONING THE SPATIAL DOMAIN INTO ZONES.

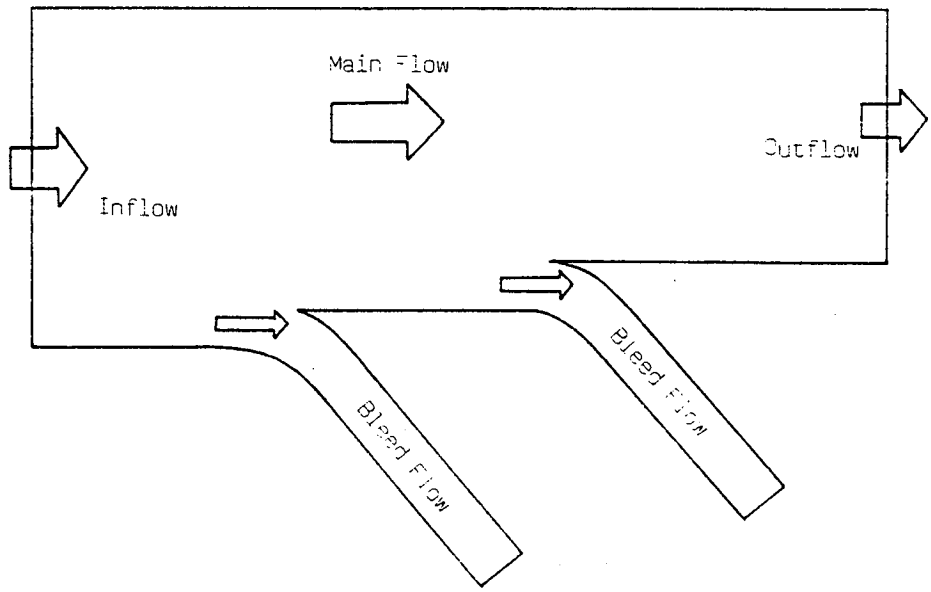
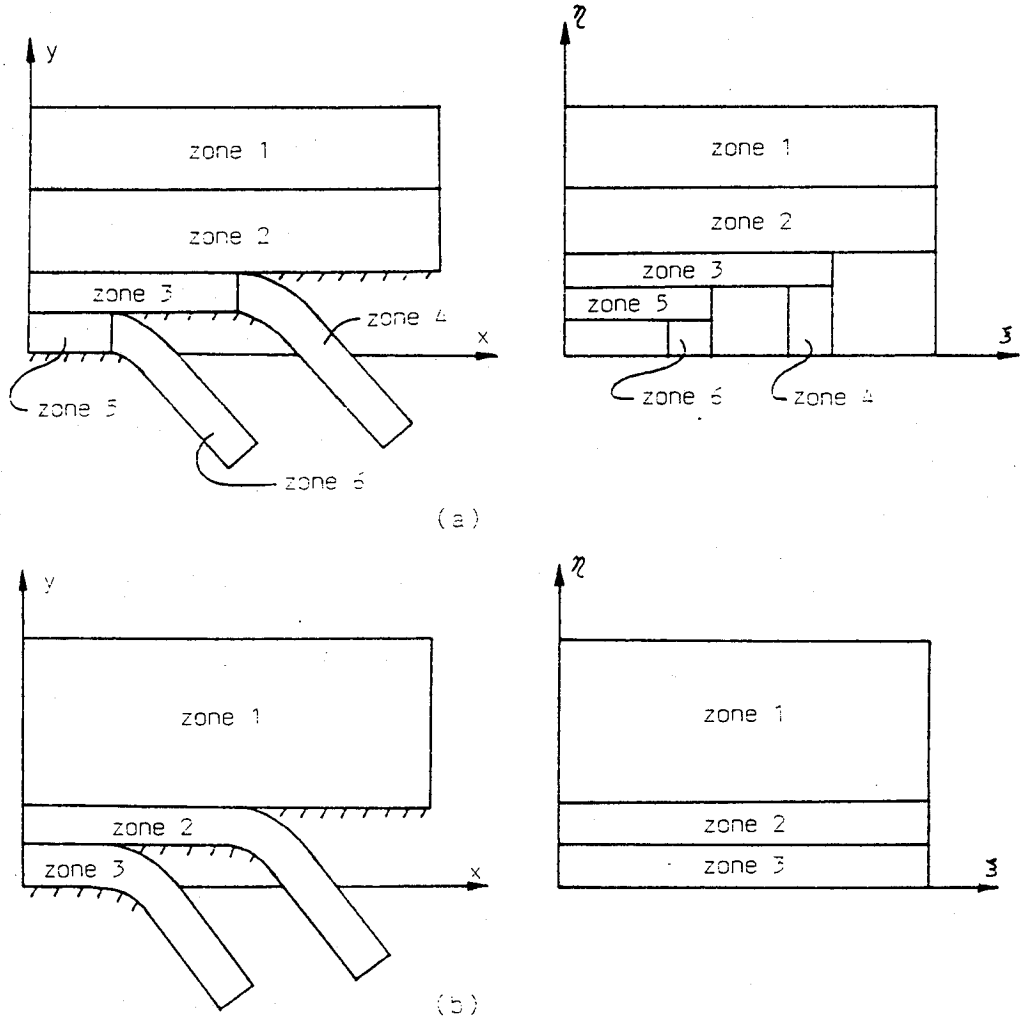


FIGURE 4-9. - TWO-DIMENSIONAL REGION SHOWING SLOTS CUT INTO WALL OF WIND TUNNEL.



(a) PARTITION WITH SIX ZONES.
(b) PARTITION WITH THREE ZONES.

FIGURE 4-10. - TWO POSSIBLE PARTITIONS FOR THE SPATIAL DOMAIN SHOWN IN FIGURE 4-9.

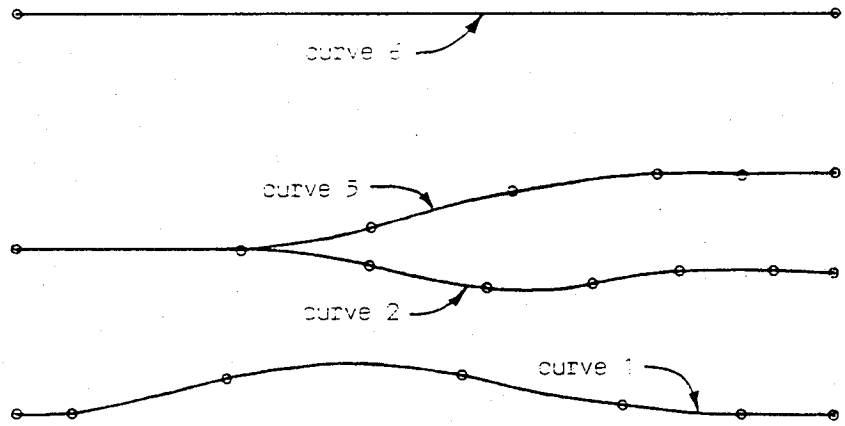


FIGURE 4-11. - APPROXIMATIONS OF CURVES 1, 2, 5 AND 6 OBTAINED BY USING PARAMETRIC TENSION SPLINE INTERPOLATION. THE CURVES ARE COMPUTER GENERATED AND NODAL POINTS ARE DENOTED BY o.

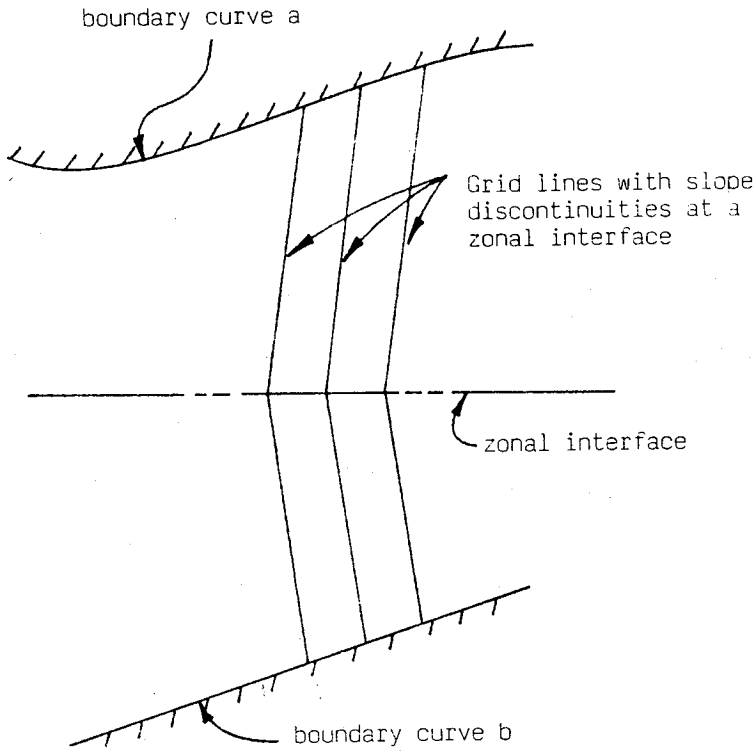
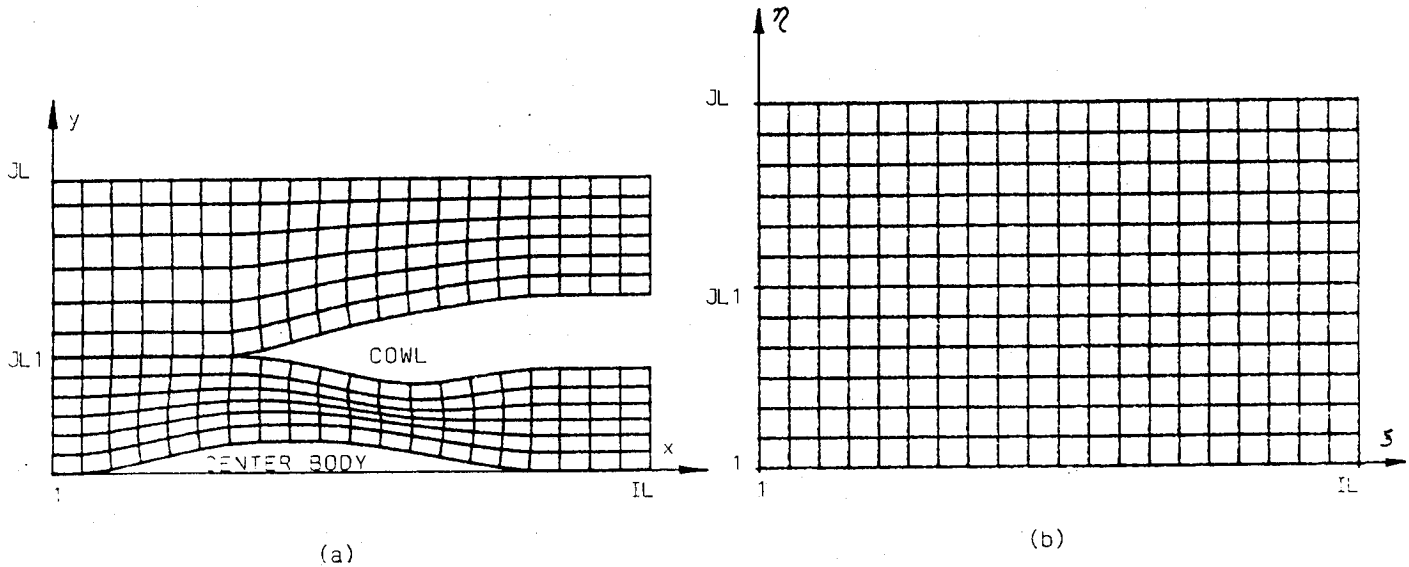


FIGURE 4-12. - AN EXAMPLE OF SLOPE DISCONTINUITIES PRESENT IN GRID LINES WHICH CROSS A ZONAL INTERFACE.



(a) IN THE SPATIAL DOMAIN.
(b) IN THE TRANSFORMED DOMAIN.

FIGURE 4-13. - GRID SYSTEM.

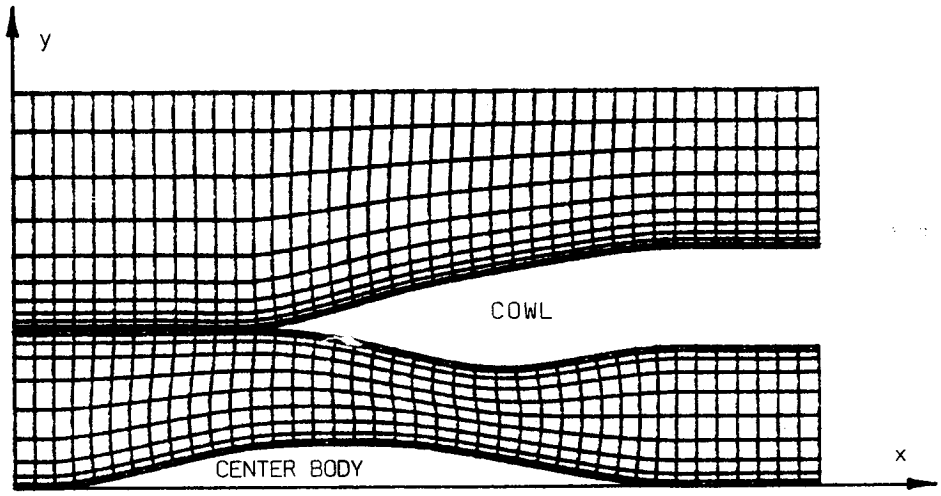


FIGURE 4-14. - GRID SYSTEM IN THE SPATIAL DOMAIN AFTER STRETCHING. GRID IS COMPUTER GENERATED.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-102453		

4. Title and Subtitle

GRID2D/3D—A Computer Program for Generating Grid Systems in Complex-Shaped Two- and Three-Dimensional Spatial Domains

Part 1: Theory and Method

5. Report Date

March 1990

6. Performing Organization Code

7. Author(s)

T.I-P. Shih, R.T. Bailey, H.L. Nguyen, and R.J. Roelke

8. Performing Organization Report No.

E-5241

10. Work Unit No.

535-05-01

9. Performing Organization Name and Address

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

11. Contract or Grant No.

13. Type of Report and Period Covered

Technical Memorandum

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Washington, D.C. 20546-0001

14. Sponsoring Agency Code

15. Supplementary Notes

T.I-P. Shih, Carnegie-Mellon University, Dept. of Mechanical Engineering, Pittsburgh, Pennsylvania 15213; R.T. Bailey, University of Florida, Dept. of Mechanical Engineering, Gainesville, Florida 32611; H.L. Nguyen and R.J. Roelke, NASA Lewis Research Center.

16. Abstract

An efficient computer program, called GRID2D/3D, has been developed to generate single and composite grid systems within geometrically complex two- and three-dimensional (2- and 3-D) spatial domains that can deform with time. GRID2D/3D generates single grid systems by using algebraic grid generation methods based on transfinite interpolation in which the distribution of grid points within the spatial domain is controlled by stretching functions. All single grid systems generated by GRID2D/3D can have grid lines that are continuous and differentiable everywhere up to the second-order. Also, grid lines can intersect boundaries of the spatial domain orthogonally. GRID2D/3D generates composite grid systems by patching together two or more single grid systems. The patching can be discontinuous or continuous. For continuous composite grid systems, the grid lines are continuous and differentiable everywhere up to the second-order except at interfaces where different single grid systems meet. At interfaces where different single grid systems meet, the grid lines are only differentiable up to the first-order. For 2-D spatial domains, the boundary curves are described by using either cubic or tension spline interpolation. For 3-D spatial domains, the boundary surfaces are described by using either linear Coon's interpolation, bi-hyperbolic spline interpolation, or a new technique referred to as 3-D bi-directional Hermite interpolation. Since grid systems generated by algebraic methods can have grid lines that overlap one another, GRID2D/3D contains a graphics package for evaluating the grid systems generated. With the graphics package, the user can generate grid systems in an interactive manner with the grid generation part of GRID2D/3D. GRID2D/3D is written in FORTRAN 77 and can be run on any IBM PC, XT, or AT compatible computer. In order to use GRID2D/3D on workstations or mainframe computers, some minor modifications must be made in the graphics part of the program; no modifications are needed in the grid generation part of the program. This technical memorandum describes the theory and method used in GRID2D/3D. Part 2 of this technical memorandum, under a separate cover, contains the computer program, GRID2D/3D, and a user's manual.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Grid generation Computational fluid mechanics Turbomachinery	Unclassified – Unlimited Subject Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	102	A06

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161